Journal of Korea Robotics Society (2016) 11(4):300-309
https://doi.org/10.7746/jkros.2016.11.4.300 ISSN: 1975-6291 / eISSN: 2287-3961

3차원 직선을 이용한 카메라 모션 추정

Motion Estimation Using 3-D Straight Lines

이 진 한[1], 장 국 현[2], 서 일 홍[†]

Jin Han Lee[1], Guoxuan Zhang[2], Il Hong Suh[†]

Abstract This paper proposes a method for motion estimation of consecutive cameras using 3-D straight lines. The motion estimation algorithm uses two non-parallel 3-D line correspondences to quickly establish an initial guess for the relative pose of adjacent frames, which requires less correspondences than that of current approaches requiring three correspondences when using 3-D points or 3-D planes. The estimated motion is further refined by a nonlinear optimization technique with inlier correspondences for higher accuracy. Since there is no dominant line representation in 3-D space, we simulate two line representations, which can be thought as mainly adopted methods in the field, and verify one as the best choice from the simulation results. We also propose a simple but effective 3-D line fitting algorithm considering the fact that the variance arises in the projective directions thus can be reduced to 2-D fitting problem. We provide experimental results of the proposed motion estimation system comparing with state-of-the-art algorithms using an open benchmark dataset.

Keywords 3D line representation, Motion estimation with lines, Visual odometry

1. 서 론

최근 빠른 속도로 픽셀단위의 컬러정보와 깊이정보를 동시에 제공하는 키넥트나 ZED와 같은 RGB-D 센서의 보급이 활발해지면서 실시간 3차원 복원, 동시 자기위치 추정기법(SLAM) 등에 대한 관심이 고조되고 로보틱스의 여러 응용분야에 적용되고 있다. 이러한 응용에 있어 가장 중요한 요소중의 하나는 바로 센서의 정확한 모션 추정인데, 이를 이용하여 각 위치에서 얻은 포인트클라우드 정보를 전체 3차원 환경지도에 정밀하게 등록할 수 있게 된다. 그동안 깊이정보를 얻을 수 있는 센서들의 모션추정을 위하여 가장 널리 사용된 알고리즘은 ICP (Iterative closest point) [1]였는데 높은 계산량과 메모리 요구량, 비교적 정확한 초기 추정값이 필요하다는 단점이 있었다.

하지만 영상의 깊이정보와 더불어 컬러정보를 함께 얻을 수 있는 RGB-D 센서의 보급이 활발해지면서 카메라의 모션을 추정하기 위하여 영상특징을 이용하는 여러 방법들이 제안되었는데, 대표적으로 특징점을 쓰는 방법들[2,3]과 특징면을 쓰는 방법들[4,5], 혹은 두 특징 모두 쓰는 방법[6]이 있다.

우리는 본 논문에서 특징점 혹은 특징면을 대체하여 직선을 특징으로 사용하는 카메라 모션 추정기법을 제안한다. 최소 3개의 독립적인 객체를 필요로 하는 특징점/면 기법과 달리 직선은 2개의 독립객체로부터 고유의 모션을 구할 수 있는 장점이 있는데 이는 곧 RANSAC (Random sample consensus) 등을 이용한 모션 검증 단계에서 샘플 조합의 수를 획기적으로 줄이는 효과를 가져온다.

Fig. 1에서 보는바와 같이 3차원 공간에서 점과 면은 3자

Received : Oct. 27. 2016; Revised : Nov. 16. 2016; Accepted : Nov. 22. 2016
※This work was supported by the Industrial Strategic Technology Development Program (10044009) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea).
†Corresponding author: Department of Electronics and Computer Engineering, Hanyang University, Wangsimni-ro 222, Seongdong-Gu, Seoul, Korea (ihsu@hanyang.ac.kr)
[1]Jin Han Lee is with the Department of Electronics and Computer Engineering, Hanyang University (jinhanlee@hanyang.ac.kr)
[2]Guoxuan Zhang is with the Department of Electronics and Computer Engineering, Hanyang University (imzgx@hanyang.ac.kr)

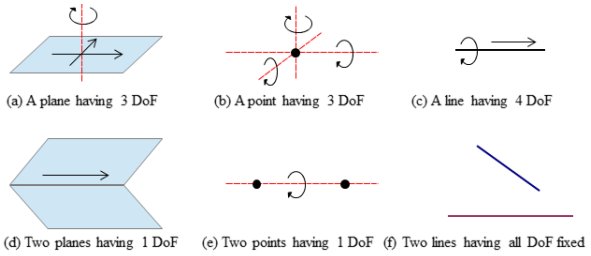


Fig. 1. Three primitives in 3-D space and their DOFs in one and two configurations

유도를 가지고, 고유의 강체(rigid body) 를 정의하기 위해서는 최소 3개의 독립적인 객체를 필요로 하게 되는데 이는 두 객체가 교차함으로써 하나의 자유도를 잃어버리기 때문이다. 하지만 이와 대조적으로 두개의 평행하지 않은 직선은 고유의 강체를 정의할 수 있는데 이는 두 직선이 한면위에 있더라도 성립한다. 3차원 공간에서 직선은 4자유도를 가지는데, 점/면에 비해 추가적으로 하나 더 있는 자유도가 두개의 직선으로부터 3차원상의 고유 강체를 정의하는 6개의 자유도를 모두 고정할 수 있게 하기 때문이다.

3차원 공간상에서의 점과 선, 면의 특성은 그것들이 포괄(span) 하는 차원의 수와 직접적으로 연관되어 있다. 점의 경우, 0차원 공간을 포괄 하는데 이는 곧 풍부한 관측 가능성과 계산상 편의성으로 연결된다. 면의 경우에는 2차원 공간을 포괄함으로써 하나의 장면에서 그 관측량이 점에 비해 현저히 작지만 대신 보다 안정적으로 관측할 수 있다는 장점이 있다. 그리고 1차원 공간을 포괄하는 직선은 관측량과 관측 안정성 측면에서 점과 면 사이에 있을 것으로 쉽게 유추 할 수 있다.

그런데, 2차원에서 직선은 하나의 방정식으로 표현되는 것과는 달리 3차원 공간상에서 직선은 그것을 매개변수화(Parameterization) 하는 주도적인 표현방법이 없다. 플뤼커좌표(Plücker coordinate), 고정점 및 방향(Anchor point plus direction), 정규직교표현(Orthonormal representation) 등의 세가지 방법이 대표적으로 쓰이는데 이들의 모션추정기법에 있어서의 자세한 평가는 보고된바가 없다. 따라서 우리는 먼저 이들 기법의 수학적 특성을 조사하고 면밀한 시뮬레이션을 통하여 모션추정기법에 있어서의 적절성을 평가하여 그 결과를 제시하고 본 논문의 모션추정기법에 이용한다.

직선을 이용한 모션추정에 있어서 본 논문은[7]의 방법을 개선하여 닫힌 형식의 해법(Closed form solution)을 유도하고 이를 이용하여 두 영상사이의 두개의 직선 대응관계로부터 모션을 추정하는 방법을 제시한다. 이 후 그 값을 초기값으로 하는 최적화 과정을 거쳐 최종 모션추정값을 얻게 되는데 여기에서 사전 시뮬레이션 검증으로부터 최적으로 평가한 정규직교표현을 사용한다.

또한 일반적으로 3차원 직선을 검출할때 주어진 포인트 클라우드에서 3차원 피팅을 하게 되는데, 우리는 주요 분산은 투영방향에서 발생한다는 점에 착안하여 3차원 피팅을 2차원 피팅으로 차원감소하는 효율적인 새로운 방법을 제안한다. 본 논문의 주요 기여사항은 다음과 같이 요약할 수 있다.

1) 면밀한 시뮬레이션을 통한 3차원 직선 표현기법의 모션추정 최적화에 있어서의 적절성 검증
2) 두쌍의 직선 대응관계를 이용한 닫힌 형태의 모션추정 해법의 제시
3) 포인트클라우드의 주요 분산을 고려한 효율적인 3차원 직선 검출 기법의 제안

2. 3차원 직선 프로세싱

2.1 3차원 기하변환

3차원 공간에서 카메라는 총 6자유도를 갖는다. i번째 카메라의 하나의 점은 $\mathbf{x}_i$로 표현되고 $\mathbf{x}_0$는 기준좌표계의 점을 표현한다. 또한 i번째 카메라에서 j번째 카메라 좌표계로의 강체변환을 위해서는 $\mathbf{T}_{ji}=[\mathbf{R}_{ji}|\mathbf{t}_{ji}]$를 사용한다.

$$\mathbf{T}_{ji}=\begin{bmatrix}\mathbf{R}_{ji} & \mathbf{t}_{ji}\\ \mathbf{0} & 1\end{bmatrix} \quad \text{with } \mathbf{R}\in SO(3),\ \mathbf{t}\in\Re^3 \tag{1}$$

따라서 두 좌표계 사이에서의 점의 변환은 $\underline{\mathbf{x}}_j=\mathbf{T}_{ji}\underline{\mathbf{x}}_i$으로 표현되는데 여기에서 $\underline{\mathbf{x}}$는 포인트 $\mathbf{x}$의 동차좌표계(Homogeneous coordinate) 표현이다. 카메라 포즈 $\mathbf{T}_{ji}$는 리 군(Lie group) SE(3) (3차원 특수 유클리드군)에 속하고 SO(3) (3차원 직교군)은 회전행렬의 리 군으로서 3차원 유클리드 공간의 회전 및 반사로 구성된다. 리 군의 연산은 행렬곱으로 정의되는데, 따라서 연속적인 카메라의 포즈는

다음과 같이 표현할 수 있다:

$$\mathbf{T}_{j0} = \mathbf{T}_{ji}\mathbf{T}_{i0} \quad \text{혹은} \tag{2}$$

$$\mathbf{T}_{ji} = \mathbf{T}_{j0}\mathbf{T}_{i0}^{-1} \tag{3}$$

2.2 3차원 직선 표현

3차원 점은 $(x,y,z)^{\mathrm{T}}$와 같이 직관적으로 표현할 수 있다. 하지만 이와 대조적으로 직선은 3차원공간상에서 4자유도를 갖기 때문에 표준화된 매개변수화 방법이 없다. 본 논문에서는 직선의 표현을 위하여 플뤼커좌표와 정규직교표현의 두가지 방법을 사용한다.

2.2.1 플뤼커좌표

Fig. 2에서 3차원 직선은 플뤼커좌표로 표현하면 6차원의 벡터 $\mathcal{L}=(\mathbf{n}^{\mathrm{T}},\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}$로 정의할 수 있는데 여기에서 $\mathbf{n}$은 원점과 $\mathcal{L}$이 이루는 평면 π의 수직벡터이고 $\mathbf{v}$는 직선의 방향벡터이다. 여기에서 $\mathbf{n}$과 $\mathbf{v}$는 단위벡터일 필요가 없고 원점으로부터 직선까지의 최단거리의 점 $\mathbf{a}$까지의 거리는 $\|\mathbf{n}\|/\|\mathbf{v}\|$로 계산할 수 있다. 하지만 Fig. 2에서 보는바와 같이 플뤼커좌표에는 일반적인 최적화 기법을 방해하는 $\mathbf{n}^{\mathrm{T}}\mathbf{v}=0$의 제약사항이 있다. 이러한 제약에도 불구하고 우리는 플뤼커좌표를 최적화단계를 제외한 모든 단계에서 사용하는데 그것은 3차원 직선을 이미지평면에 투영할때와 새로 관측된 직선을 초기화할때 편리하기 때문이다. i번째 카메라에서 j번째 카메라로의 플뤼커좌표로 표현된 직선의 기하변환은 다음과 같이 선형연산으로 정의 된다.

$$\begin{bmatrix}\mathbf{n}_j\\ \mathbf{v}_j\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{ji} & [\mathbf{t}_{ji}]\\ \mathbf{0} & \mathbf{R}_{ji}\end{bmatrix}\begin{bmatrix}\mathbf{n}_i\\ \mathbf{v}_i\end{bmatrix} \tag{4}$$
$$\mathcal{L}_j = \mathbf{H}_{ji}\mathcal{L}_i$$

여기에서 $[\mathbf{t}_{ji}]_{\times}$는 $\mathbf{t}_{ji}$의 Skew-symmetric matrix 이다.

2.2.2 정규직교표현

직선의 플뤼커좌표 표현 $\mathcal{L}=(\mathbf{n}^{\mathrm{T}},\mathbf{v}^{\mathrm{T}})^{\mathrm{T}}$로부터 정규직교표현 $(\mathrm{U},\mathrm{W})\in SO(3)\times SO(2)$는 QR분해를 사용하여 다음과 같이 구할 수 있다.

$$[\mathbf{n}|\mathbf{v}] = \mathrm{U}\begin{bmatrix}\omega_1 & \\ & \omega_2\end{bmatrix}, \text{ and set } \mathrm{W} = \begin{bmatrix}\omega_1 & -\omega_2\\ \omega_2 & \omega_1\end{bmatrix} \tag{5}$$

Fig. 2에 보인바와 같이 U와 W는 각각 3차원과 2차원의 회전행렬이다. 따라서 정규직교표현은 3차원 공간상에서 직선을 표현하는데 필요한 최소갯수인 4개의 매개변수로 $\mathbf{P}^{\mathrm{T}}=(\theta,\ \theta)^{\mathrm{T}}$와 같이 표현할 수 있고, 직선 (U,W)의 업데이트는 $\mathrm{U}\leftarrow\mathrm{U}\mathbf{R}(\theta)$, $\mathrm{W}\leftarrow\mathrm{U}\mathbf{R}(\theta)$와 같이 할 수 있다. 정규직교표현으로부터 플뤼커좌표으로의 변환은 다음과 같이 정의되는데,

$$\mathcal{L} = (\omega_1\mathbf{u}_1^{\mathrm{T}}, \omega_2\mathbf{u}_2^{\mathrm{T}})^{\mathrm{T}} \tag{6}$$

여기에서 $\mathbf{u}_i$는 U의 i번째 컬럼이다.

추가적으로, 다른 관련 연구들에서 3차원 직선을 표현하는데 많이 적용되는 방법으로 고정점 및 방향의 방법이 있는데, 이는 직선을 원점과 직선사이의 최단거리점 $\mathbf{a}$와 방향벡터 $\mathbf{v}$, 즉 $(\mathbf{a},\mathbf{v})^{\mathrm{T}}\in\Re^6$으로 표현한다. 표현에 있어서 직관적이고 두 벡터사이에 제약사항이 없다는 장점이 있지만, 3차원 직선을 표현하는데 있어서 필요한 최소차원의 수인 4보다 더 많은 총 6차원의 벡터로 이루어져 있기 때문에 최적화에서 불리하게 작용할 수 있음을 짐작할 수 있다. 우리는 최적화 시뮬레이션을 통해 두 표현기법을 비교하였고 우리의 직관대로 정규직교표현이 최적화에 있어서 훨씬 효과적임을 검증하였다. 따라서 본 논문의 최적화단계에서

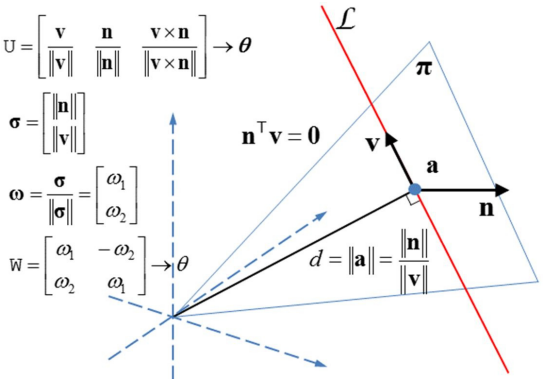


Fig. 2. Line representations in 3-D space

정규직교표현을 적용하고 시뮬레이션 결과는 뒤의 실험섹션에서 제시한다.

2.3 RGB-D 영상에서의 3차원 직선의 검출

3차원 직선의 검출은 먼저 주어진 RGB영상에서 2차원 직선을 검출하는 것으로 시작하는데 본 논문에서는 이를 위하여[7]에서 제안된 방법을 이용한다. Fig. 3에서 보이는 바와 같이 2차원 직선 l위의 총 k개의 픽셀에 대하여 깊이값을 이용하여 3차원 공간상의 포인트클라우드 $\mathrm{S} \in \Re^{3\times k}$를 정의하고 단순하게는 여기에 바로 3차원 직선피팅 알고리즘을 적용하여 원하는 3차원 직선을 계산 할 수 있다.

하지만, Fig. 3에서 보이는 바와 같이, 이미 2차원 직선 검출단계에서 직선피팅이 이루어진 경우 S의 주된 분산은 평면 π상에서 투영방향에 존재함을 알 수 있다. 따라서 본 논문에서는 S에 바로 3차원 직선피팅을 적용하지 않고, 먼저 평면 좌표계에 S를 재투영한 S'을 계산하고 여기에 2차원 직선피팅을 적용하여 2차원 직선의 방정식을 계산한 뒤 다시 역변환을 통해 v를 얻는다.

이를 위하여 먼저 평면 π의 수직벡터는 $\mathbf{n}=\mathbf{e}\times\mathbf{f}$로 쉽게 구할 수 있다. 여기에서 e와 f는 l의 양끝점에 깊이값을 이용하여 투영한 점으로서 깊이값이 정확하지 않더라도 두 벡터의 외적으로 구하는 n의 결과에는 큰 영향을 미치지 않음을 알 수 있다. 그리고 포인트클라우드에서 임의의 한 점 g를 선택하면 S의 S'으로의 회전변환 R을 다음과 같이 구할 수 있다.

$$\mathrm{R}=\left[\frac{(\mathbf{g}\times\mathbf{n})^{\mathrm{T}}}{\|\mathbf{g}\times\mathbf{n}\|}\ \frac{\mathbf{g}^{\mathrm{T}}}{\|\mathbf{g}\|}\ \frac{\mathbf{n}^{\mathrm{T}}}{\|\mathbf{n}\|}\right]^{\mathrm{T}},\ \mathrm{S}'=\mathrm{RS} \qquad (7)$$

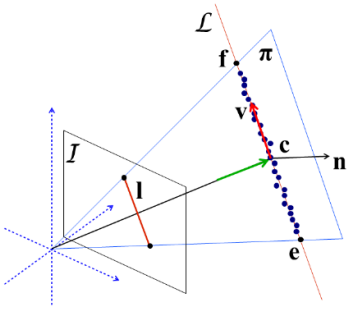


Fig. 3. 3-D line extraction

2차원 직선피팅을 위하여 S'의 원소 S'_{ij}를 이용하여 행렬 A와 벡터 b를 다음과 같이 정의하면,

$$\mathrm{A}=\begin{bmatrix} S'_{11} & 1 \\ \vdots & \vdots \\ S'_{1k} & 1 \end{bmatrix},\ \mathrm{b}=\begin{bmatrix} \mathrm{S}'_{21} \\ \vdots \\ \mathrm{S}'_{2k} \end{bmatrix} \qquad (8)$$

일반적인 선형 최소자승법을 이용하여 $\mathrm{Al}'=\mathrm{b}$를 최적화하는 l'을 계산할 수 있는데 여기에서 l'은 평면 π에서 정의된 2차원 직선의 방정식이다. 이제 양끝점 $(S'_{11}, S'_{21})^{\mathrm{T}}$과 $(S'_{1k}, S'_{2k})^{\mathrm{T}}$를 피팅된 l'에 투영하여 $\mathbf{e}'$과 $\mathbf{f}'$을 얻고 e와 f를 다음과 같이 업데이트 한다.

$$\mathbf{e}=\mathrm{R}^{\mathrm{T}}(\mathbf{e}';0),\ \mathbf{f}=\mathrm{R}^{\mathrm{T}}(\mathbf{f}';0)$$

마지막으로 최종 v는 f−e로 얻게 된다.

3. 3차원 직선을 이용한 모션 추정

3.1 두쌍의 직선 대응관계를 이용한 모션 추정

Fig. 4에서 보이는 바와 같이 두개의 3차원 직선 $\mathcal{L}^{\mathrm{I}}$과 $\mathcal{L}^{\mathrm{II}}$가 두 카메라 1, 2에서 각각 $\{\mathcal{L}_1^{\mathrm{I}}, \mathcal{L}_1^{\mathrm{II}}\}$와 $\{\mathcal{L}_2^{\mathrm{I}}, \mathcal{L}_2^{\mathrm{II}}\}$로 관측되면 다음의 단계를 통하여 닫힌 형태의 해법으로 모션을 추정할 수 있다. 먼저 Horn의 방법[8,9]을 이용하여 쿼터니언 회전성분 $\mathbf{q}_{21}$을 구하고 플뤼커좌표의 기하변환 식으로부터 $\mathbf{t}_{21}$를 구한다.

더 자세히 설명하면, 먼저 관측된 두 직선의 대응관계를 다음과 같이 정의할 때,

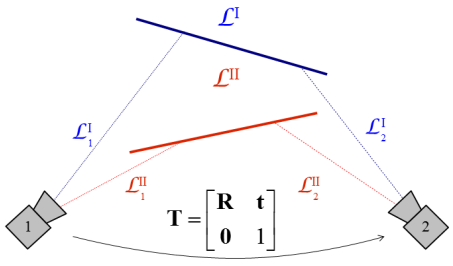


Fig. 4. 3D lines $\mathcal{L}^{\mathrm{I}}$and $\mathcal{L}^{\mathrm{II}}$are observed from two camera poses as $\{\mathcal{L}_1^{\mathrm{I}}, \mathcal{L}_1^{\mathrm{II}}\}$ and $\{\mathcal{L}_2^{\mathrm{I}}, \mathcal{L}_2^{\mathrm{II}}\}$

$$\mathcal{L}_1^{\mathrm{I}} = \begin{bmatrix} \mathbf{n}_1^{\mathrm{I}} \\ \mathbf{v}_1^{\mathrm{I}} \end{bmatrix},\ \mathcal{L}_1^{\mathrm{II}} = \begin{bmatrix} \mathbf{n}_1^{\mathrm{II}} \\ \mathbf{v}_1^{\mathrm{II}} \end{bmatrix}$$
$$\mathcal{L}_2^{\mathrm{I}} = \begin{bmatrix} \mathbf{n}_2^{\mathrm{I}} \\ \mathbf{v}_2^{\mathrm{I}} \end{bmatrix},\ \mathcal{L}_2^{\mathrm{II}} = \begin{bmatrix} \mathbf{n}_2^{\mathrm{II}} \\ \mathbf{v}_2^{\mathrm{II}} \end{bmatrix}$$

$\mathbf{q}_{21}$을 구하기 위하여 먼저 모든 $\mathbf{v}$를 정규화(Normalization) 하고 Horn의 방법을 적용하여 $\mathbf{q}_{21}$를 계산하여 이를 회전행렬 $\mathbf{R}_{21}$로 변환한다. 이후 식 (4)로부터 다음과 같이 전개할 수 있다.

$$\mathbf{n}_2^i = \mathbf{R}_2\mathbf{n}_1^i + [\mathbf{t}_{21}]_\times \mathbf{R}_{21}\mathbf{v}_1^i \tag{9}$$

$$[\mathbf{t}_{21}]_\times \mathbf{R}_{21}\mathbf{v}_1^i = \mathbf{n}_1^i - \mathbf{R}_{21}\mathbf{n}_1^i \tag{10}$$

$$[-\mathbf{R}_{21}\mathbf{v}_1^i]_\times \mathbf{t}_{21} = \mathbf{n}_2^i - \mathbf{R}_{21}\mathbf{n}_1^i \tag{11}$$

여기에서 $\mathbf{A}^i = [-\mathbf{R}_{21}\mathbf{v}_1^i]_\times$, $\mathbf{b}^i = \mathbf{n}_2^i - \mathbf{R}_{21}\mathbf{n}_1^i$를 정의하고 두 대응관계를 모두 이용하여 다음과 같이 정의하면,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{\mathrm{I}} \\ \mathbf{A}^{\mathrm{II}} \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} \mathbf{b}^{\mathrm{I}} \\ \mathbf{b}^{\mathrm{II}} \end{bmatrix}$$

일반적인 선형 최소자승법을 이용하여 $\mathbf{A}\mathbf{t}_{21} = \mathbf{b}$를 최적화하는 $\mathbf{t}_{21}$를 계산할 수 있다.

3.2 비선형 최소자승법을 이용한 모션 최적화

비선형 최소자승법(Nonlinear least squares)을 적용하기 위해서는 먼저 초기 모션 추정값을 구해야 한다. 이를 위하여, 두 RGB-D 영상 사이에서 여러개의 직선 대응관계가 주어지면 RANSAC을 이용한다. 반복적으로 랜덤하게 두 개의 대응관계를 선택하고 그로부터 계산한 모션에 부합하는 인라이어(Inlier) 대응관계를 구하여 총 관측오차가 가장 적은 모션을 선택하는 것이다.

RANSAC 각 이터레이션(Iteration)에서 관측오차를 계산하고 인라이어를 결정하는 것은 다음과 같이 할 수 있다. 먼저, 두개의 직선 대응관계로부터 계산한 모션과 식 (4)를 이용하여 카메라 프레임 1의 모든직선을 카메라 프레임 2의 좌표계로 변환한다. 하나의 대응관계를 예로들면, $\{\mathcal{L}_1, \mathcal{L}_2\}$를 카메라 1, 2에서의 대응관계라고 할때 $\mathcal{L}_1$의 계산한 모션에 의한 카메라 2의 좌표계에서의 예측값은 $\mathcal{L}'_2$=H$\mathcal{L}_1$로 계산할 수 있고 관측오차는 카메라 2에서 관측한 직선 $\mathcal{L}_2$의 양끝점과 $\mathcal{L}'_1$사이의 수직거리로서 계산할 수 있다. 그리고 이 관측오차가 일정값보다 작은 대응관계를 인라이어로 판정하는데 이렇게 여러번의 이터레이션을 수행하여 모든 인라이어들의 총 관측오차가 가장 작은 가설 모션을 선택하게 된다.

이제 이렇게 구한 초기 모션 추정값과 인라이어 대응관계를 이용하여 비선형 최소자승법을 통해 모션을 더 정밀화 할 수 있다. 본 논문은 이를 위하여 가우스-뉴턴 방법의 증강된 버젼인 Levenberg-Marquardt (LM) 알고리즘을 사용한다[10,11]. 각 LM 이터레이션에서 업데이트 벡터 $\boldsymbol{\delta}$는 다음의 정규방정식(Normal equation)의 해를 구함으로써 계산할 수 있다:

$$(\mathbf{J}^{\mathrm{T}}\boldsymbol{\Lambda}\mathbf{J} + \lambda\mathbf{I})\boldsymbol{\delta} = -\mathbf{J}^{\mathrm{T}}\boldsymbol{\Lambda}\boldsymbol{\epsilon} \tag{12}$$

여기에서 λ는 수렴정도에 따라 업데이트 스텝(Update step)을 결정짓는 스칼라값이고 $\boldsymbol{\epsilon}$은 앞서 설명한 6차원의 관측오차(한 끝점에 3차원) 벡터, $\mathbf{J}$는 관측오차의 상태 파라미터(State parameters)에 대한 자코비안 행렬(Jacobian matrix), $\boldsymbol{\Lambda}$는 정보행렬(Information matrix) 이다. 본 논문에서는 이러한 정규방정식의 해를 구하기 위하여 구글의 Ceres Solver[12]를 사용하였다.

인접한 두 프레임사이의 모션을 알 수 있으면 연속하여 들어오는 프레임들로 카메라 포즈와 랜드마크 그래프를 형성하고 그 안에서 일정크기의 움직이는 윈도우를 설정하여 윈도우 안의 카메라포즈와 랜드마크를 함께 최적화할 수 있는데 이러한 방법을 지역적 광속조정(Local Bundle Adjustment, LBA)라 한다. 크지 않은 계산비용으로 더 정확한 결과를 얻을 수 있기 때문에 대부분의 영상기반 주행기록계(Visual odometry) 방법들이 LBA를 적용하고 있고, 따라서 본 논문에서도 LBA를 이용하여 카메라 모션을 더 정확하게 최적화한다.

Fig. 5는 LBA가 적용되는 카메라 포즈와 랜드마크 그래프를 보여주는데 빨간색 직선은 랜드마크를, 녹색 카메라는

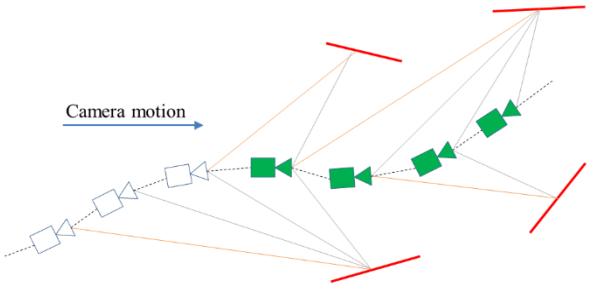


Fig. 5. Camera pose and landmark graph for LBA

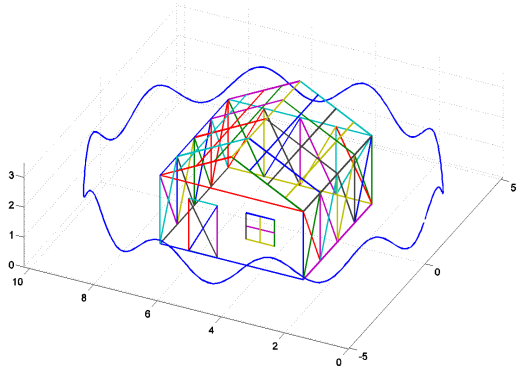


Fig. 6. Simulated 3-D house model and trajectory

최적화 되는 카메라, 흰 카메라는 고정된 카메라를 나타낸다. 즉 공통의 랜드마크 관측을 가지는 네개의 카메라 포즈(무빙윈도우 사이즈 4)에 대하여 랜드마크와 함께 최적화를 수행하게 되는 것이다. 뒤의 실험장에서 본논문이 제안하는 방법에 LBA를 적용하기 전과 적용후의 두 경우 모두에 대해서 실험을 진행하고 그 결과를 제시한다.

4. 실험결과

4.1 시뮬레이션을 통한 직선표현방법의 최적화 검증

본 논문에서 제안하는 방법은 3차원 직선을 다루는데에 대부분 플뤼커좌표를 사용하는데 앞에서 설명한바와 같이 해당 표현방법은 $\mathbf{n}$과 $\mathbf{v}$가 서로 수직해야 한다는 제약사항 탓에 일반적인 최적화기법에 적용하는데 무리가 있다. 따라서 대체수단을 찾아야 하는데 이를 위하여 일반적으로 고려되는 다른 직선 표현 방법으로는 2.2.2에서 설명한 정규직교표현과 고정점 및 방향 방법이 있다.

이 장에서는 위의 두 표현방법을 LBA 시뮬레이션에 적용하여 최적화 기법에서의 적절성을 평가한다. Fig. 6은 이 시뮬레이션에서 사용한 랜드마크모델과 카메라 궤적을 보이는데, 집 형태의 골격구조 모델을 돌면서 총 400개의 포즈를 설정하였고, 각 포즈에서 가상의 카메라를 이용하여 직선을 관측하여 LBA에 적용 하였다. 다양한 셋팅으로 시뮬레이션을 수행하였는데, 직선을 관측하는데 있어서 노이즈를 추가하는 관측 표준편차 σ_o={0.2, 0.4, 0.6, 0.8, 1.0}, LM의 최대 제한 이터레이션 수 N={10, 1000}, LBA에서 함께 고려하는 프레임 수를 결정짓는 무빙 윈도우의 크기 W={5, 10, 20, 40}으로 설정하였다.

Table 1은 시뮬레이션 결과를 보이는데 여기에서 ϵ_s는 총 관측오차합, ϵ_a는 평균 관측오차, τ는 평균 수렴 이터레이션수, t는 총 소요시간을 나타낸다. 결과에서 알 수 있듯이, 우리의 직관대로, 3차원상에서 직선을 표현하는데 필요한 파라미터의 최소 갯수인 4개의 원소를 사용하는(4차원 벡터) 정규직교표현이 모든 경우에서 6차원의 고정점 및 방향 표현방법을 압도하는 것을 볼 수 있다. 따라서 우리는 본논문의 최적화 단계에서 정규직교표현을 직선표현방법으로 사용한다.

4.2 공개 데이터셋을 이용한 최신 기술들과의 비교

이 장에서는 본 논문에서 제안하는 모션추정 방법을 검증하기 위하여 현재 학계에 보고된 최신기술들과 공개 데이터셋을 이용한 비교실험을 진행하여 그 결과를 제시한다. 데이터셋으로는 TUM RGB-D 벤치마크 데이터셋[13]의 Freiburg2/desk를 사용하였고 정확도 계산을 위해서는 상대포즈오차(Relative pose error)를 이용하였다. 상대포즈오차는 인접한 두 프레임사이에서 추정된 모션에서 이동값(Translation)과 회전값(Rotation) 을 추출하고 데이터셋에서 제공하는 실측값(Ground truth)과 비교하여 구하게 되는데, 전역일관성(Global consistency)을 고려하지 않는 주행기록계방법의 평가에 적합하다.

앞에서 언급한대로 RGB-D 영상이 주어지면 먼저 RGB 영상에서[7]에서 제안된 방법을 이용하여 2차원 직선세그먼트를 검출하고 깊이영상을 이용하여 각 직선세그먼트를 3차원으로 투영한 뒤 2.3에서 제안한 방법으로 3차원 직선을 계산하였다. 두 영상사이의 직선의 대응관계는, 먼저 각 직

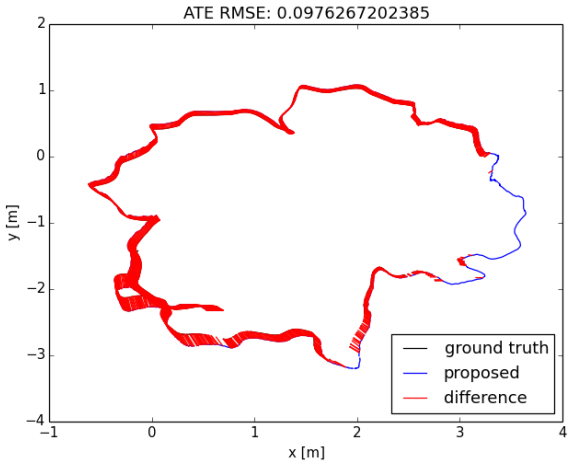
Table 1. Simulation Results

		Anchor Point Plus Direction, $N=10$															
		W															
		5				10				20				40			
σ_o		ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t
	0.2	253.97	0.634	9.93	39.73	309.63	0.774	9.92	49.47	85.36	0.213	9.88	74.83	10.76	0.026	9.06	134.40
	0.4	231.81	0.579	9.93	39.07	64.65	0.161	9.88	48.55	9.64	0.024	9.73	72.89	9.86	0.024	9.26	136.97
	0.6	125.09	0.321	9.91	39.98	116.54	0.291	9.92	49.38	30.93	0.077	9.80	74.04	3.49	0.008	9.37	139.03
	0.8	148.22	0.370	9.92	39.25	111.17	0.277	9.87	47.79	8.00	0.020	9.63	73.33	1.89	0.004	8.78	133.55
	1	26.17	0.065	9.90	38.31	9.72	0.024	9.89	47.60	7.55	0.018	9.64	72.92	8.91	0.022	9.06	136.13

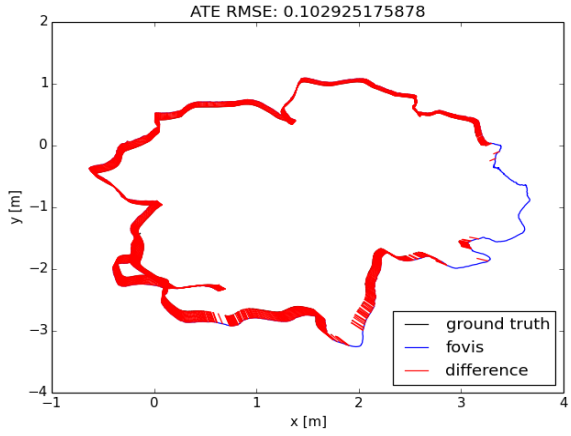
		Anchor Point Plus Direction, $N=1000$															
		W															
		5				10				20				40			
σ_o		ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t
	0.2	5.71	0.014	217.71	265.27	3.19	0.007	105.28	237.21	2.31	0.005	50.34	251.94	1.18	0.002	23.61	270.59
	0.4	10.91	0.027	310.20	367.70	13.31	0.033	111.80	250.54	5.29	0.013	52.23	258.20	2.25	0.005	20.46	245.29
	0.6	168.62	0.421	316.79	376.40	222.30	0.555	105.79	234.45	24.08	0.060	29.50	137.35	3.50	0.008	13.53	166.20
	0.8	146.98	0.367	320.91	379.45	105.72	0.264	107.93	243.37	5.94	0.014	32.56	170.06	1.92	0.004	14.03	175.66
	1	23.81	0.059	261.70	315.30	15.18	0.037	86.48	202.65	7.54	0.018	34.42	180.00	8.85	0.002	14.25	181.19

		Orthonormal Representation, $N=10$															
		W															
		5				10				20				40			
σ_o		ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t
	0.2	5.88	0.014	2.34	33.04	3.55	0.008	2.21	36.98	2.32	0.005	1.46	42.46	1.16	0.002	1.14	55.70
	0.4	10.79	0.026	3.97	34.93	13.16	0.032	3.38	39.51	5.19	0.012	2.27	47.00	2.28	0.005	2.10	66.59
	0.6	16.03	0.040	5.70	37.99	9.28	0.023	4.27	42.27	4.10	0.010	3.19	51.95	3.49	0.008	2.28	67.75
	0.8	15.75	0.039	6.81	38.44	12.94	0.032	4.85	43.22	5.71	0.014	3.48	52.91	1.87	0.004	3.06	77.06
	1	28.06	0.071	7.49	39.12	15.64	0.039	5.35	44.41	7.55	0.018	4.05	55.40	8.89	0.022	3.24	78.33

		Orthonormal Representation, $N=10$															
		W															
		5				10				20				40			
σ_o		ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t	ϵ_s	ϵ_a	τ	t
	0.2	5.53	0.013	2.37	31.96	3.55	0.008	2.24	35.85	2.32	0.005	1.47	41.45	1.16	0.002	1.14	54.40
	0.4	10.86	0.027	4.20	34.22	13.16	0.032	3.36	38.30	5.19	0.012	2.30	45.90	2.28	0.005	2.13	65.20
	0.6	16.09	0.040	5.98	36.30	9.26	0.023	4.48	42.21	4.10	0.010	3.28	50.89	3.49	0.008	2.38	66.81
	0.8	15.70	0.039	7.48	38.45	12.94	0.032	5.23	43.54	5.70	0.014	3.60	52.71	1.87	0.004	3.15	77.86
	1	28.47	0.071	8.52	41.05	15.74	0.039	6.00	46.64	7.56	0.018	4.26	56.99	8.89	0.022	3.32	79.59



(a) Result from the proposed system



(b) Result from Fovis[15]

Fig. 7. Resulting trajectories and Absolute Trajectory Error to the ground truth from (a) the propoAsed system and (b) Fovis[15]

Table 2 Experimental Comparison Results

Methods	$\tilde{x}$[m]	σ_x	$\tilde{\theta}$[°]	σ_θ
DVO	0.024	0.012	0.982	0.512
Realtime NDT	0.024	0.015	1.228	0.735
CCNT	0.020	0.012	0.762	0.457
DEMO	0.023	0.015	0.997	0.628
Fovis	0.012	0.007	0.526	0.307
Ours	0.013	0.008	0.565	0.296
Ours + LBA	0.012	0.007	0.542	0.287

선에 직선특징 디스크립터 Mean Standard-deviation Line Descriptor (MSLD)[14]를 적용하여 직선특징을 기술하고 최근린거리비율(Nearest Neighbor Distance Ratio, NNDR) 기법을 이용하여 구하였다.

비교한 기술들은 Fovis[15], DVO[16], Realtime NDT[17], CCNY[18], DEMO[19] 등으로서 모두 RGB 영상에서 영상 특징점을 추출하여 대응관계를 구하고 깊이정보를 이용하여 초기모션을 구하며 모션 최적화 과정을 통해 최종 모션 추정 값을 얻는 방법들이다.

Table 2에서 결과를 확인할 수 있는데 $\tilde{x}$는 미터 단위의 위치오차 평균(Mean of translational error), σ_x는 위치오차의 표준편차, $\tilde{\theta}$는 각도 단위의 회전오차 평균(Mean of rotational error), σ_θ는 회전오차의 표준편차를 나타낸다. Table에서 보이는 바와 같이, 모든 경우에서 본 논문에서 제안한 방법(Ours, Ours + LBA)이 Fovis와 비슷한 정확도 성능을 가지고 나머지 방법들 보다는 좋은 결과를 내는 것을 확인 할 수 있다. 회전오차평균은 Fovis가 좋은 반면 회전오차의 표준편차는 본논문에서 제안한 방법이 좋았다. 또한 Fig. 7은 제안하는 방법과 Fovis로 부터 얻은 궤적결과를 실제값과 비교하는 절대위치오차(Absolute Trajectory Error)를 보이는데 제안하는 방법으로 부터 계산한 평균제곱근오차(Root Mean Square Error)가 약 0.0976인 반면 Fovis는 약 0.1029로서 제안하는 방법이 약 5% 정도 우수한 성능을 보이는 것을 확인 할 수 있다. Table 2에서 본논문이 제안하는 방법들 외 다른 방법들의 실험결과는 본 논문에서 사용한 것과 같은 데이터셋(Freiburg2/desk)를 이용하여 여러 RGB-D visual odometry 방법들을 비교한 논문인[20]에서 인용하였다.

5. 결 론

본 고에서는 3차원 직선을 이용한 카메라 모션 추정기법을 제안하였다. 이를 위하여 먼저 모션추정 최적화에 있어서 적합한 직선표현방법을 면밀한 시뮬레이션을 통하여 검증하였고 두개의 직선 대응관계로부터 초기모션을 추정하는데 있어서 닫힌 형태의 해법을 제시하였다. 또한 3차원 직선을 검출하는데 있어서 주요 분산을 고려하여 2차원 직선피팅으로 문제를 줄임으로써 효율적이면서도 효과적인 새로운 방법을 제안하였다. 제안한 모션추정방법을 검증하기 위하여 현재 학계에 보고 된 최신기술들과 공개 데이터

셋을 이용하여 비교 실험 하였고 경쟁력 있는 결과를 얻을 수 있었다.

References

[1] P.J. Besl and N.D. McKay, "A method for registration of 3-D shapes," IEEE Trans. Pattern Anal. Mach. Intell. (PAMI), vol.14, no.2, pp.239-256, February, 1992.

[2] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in International Symposium on Experimental Robotics (ISER), 2010.

[3] H. Strasdat, A. Davison, J. Montiel, and K. Konolige, "Double Window Optimisation for Constant Time Visual SLAM," in IEEE International Conference on Computer Vision (ICCV), 2011.

[4] J. Weingarten and R. Siegwart, "3D SLAM using planar segments," in IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS), pp.3062-3067, 2006.

[5] K. Pathak, A. Birk, N. Vaˇskeviˇcius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-D mapping," IEEE Trans. Robotics (T-RO), vol.26, no.3, pp.424-441, June, 2010.

[6] Y. Taguchi, Y. Jian, S. Ramalingam, and C. Feng, "Point-Plane SLAM for Hand-Held 3D Sensors," in IEEE International Conference on Robotics and Automation (ICRA), 2013.

[7] J.H. Lee, G. Zhang, J. Lim, and I.H. Suh, "Place Recognition using Straight Lines for Vision-based SLAM," in IEEE International Conference on Robotics and Automation (ICRA), 2013.

[8] B.K.P. Horn, "Closed-form solution of absolute orientation using unit quaternions," Journal of Optical Society of America A, vol.4, no.4, pp.629-642, 1987.

[9] K. Kanatani, "Unbiased estimation and statistical analysis of 3-D rigid motion from two views," IEEE Trans. Pattern Anal. Mach. Intell. (PAMI), vol.15, no.1, pp.37-50, June, 1993.

[10] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," Quarterly of Applied Mathematics, vol.2, no.2, pp.164-168, 1944.

[11] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," Journal of the Society for Industrial and Applied Mathematics, vol.11, no.2, pp.431-441, 1963.

[12] S. Agarwal and K. Mierle, "Ceres Solver: Tutorial & Reference," Google Inc., http://code.google.com/p/ceres-solver/.

[13] J. Sturm, W. Burgard, and D. Cremers, "Evaluating Egomotion and Structure-from-Motion Approaches Using the TUM RGB-D Benchmark," in IEEE/RJS International Conference on Intelligent Robot (IROS), 2012.

[14] Zhiheng WANG, Fuchao WU, Zhanyi HU, "MSLD: A Robust Descriptor for Line Matchingm," Pattern Recognition, vol.42, no.5, pp.941-953, 2009.

[15] AS Huang and Abraham Bachrach, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in International Symposium on Robotics Research, pp.1-16, 2011.

[16] Christian Kerl, Jurgen Sturm, and Daniel Cremers, "Robust Odometry Estimation for RGB-D cameras," In IEEE International Conference on Robotics and Automation (ICRA), pp.3748-3754, May, 2013.

[17] Henrik Andreasson and Todor Stoyanov, "Real Time Registration of RGB-D Data Using Local Visual Features and 3D-NDT Registration," in IEEE International Conference on Robotics and Automation (ICRA) Workshop on Semantic Perception, Mapping and Exploration (SPME), 2012.

[18] Ivan Dryanovski and Roberto G. Valenti, "Fast Visual Odometry and Mapping from RGB-D Data," In IEEE International Conference on Robotics and Automation (ICRA), pp.2305-2310, May, 2013.

[19] Ji Zhang, Michael Kaess, and Sanjiv Singh, "Realtime Depth Enhanced Monocular Odometry," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), September, 2014.

[20] Zheng Fang, Yu Zhang, "Experimental Evaluation of RGB-D Visual Odometry Methods," International Journal of Advanced Robotic Systems (IJARS), December, 2015.

이 진 한

2010 한양대학교 전자컴퓨터공학부(공학사)
2012 한양대학교 전자컴퓨터통신공학과(공학석사)
2012~현재 한양대학교 전자컴퓨터통신공학과 박사과정

관심분야: 컴퓨터비전, 기계학습, SLAM

서 일 홍

1979 한국과학기술원 전자공학과(공학석사)
1982 한국과학기술원 전자공학과(공학박사)
1985~현재 한양대학교 융합전자공학부 교수

관심분야: 인공지능, 기계학습, 컴퓨터비전

장 국 현

2003 한양대학교 정보처리공학과(공학석사)
2012 한양대학교 전자컴퓨터통신공학과(공학박사)

관심분야: 컴퓨터비전, 기계학습, SLAM