2001년도 대한전기학회/대한전자공학회 시스템및제어분야 합동 추계학술대회 논문집 (2001.11.24)

비젼시스템를 이용한 수중로봇의 위치추정

김진석*, 김흥수**, 조병학*, 김준홍*, 신창훈*, 김석곤*
한전전력연구원*, 한국과학기술원**

Pose Estimation of Underwater Robot using Vision System

Jin-Seok Kim*, Heung-Soo Kim**, Byung-Hak Cho*, Joon-Hong Kim*, Chang-Hoon Shin*, Seok-Gon Kim*
KEPRI*, KAIST**

Abstract - Nuclear regulation requires a periodic visual test for inside structures of reactor to guarantee safe operation of nuclear power plant. However, existing visual test, which is proceeded manually, needs lots of time and labor. Even more, test workers should be exposed in radioactive environment during the test. An underwater robot system has being studied for more efficient and safer test. The position and pose estimation are important issue for the movement control of the robot. An algorithm was presented in this paper, which estimate the location and pose of the underwater robot clearly using vision system.

1. 서 론

경수로 원자로의 내부 격벽(Baffle)은 여러 조각으로 구성되어 있으며, 이를 포머(Former)에 체결하기 위한 볼트 수 또한 약 천여 개에 이르므로, 이 포머볼트의 신속한 육안검사를 위해서는 자동화된 방법을 필요로 한다. 현재 이용되는 검사 방법은 검사자가 긴 봉 끝에 카메라를 설치하여 직접 모니터링하며 수작업으로 검사하는 방법과 웨스팅하우스사의 Supreme과 같은 대형 구조물을 원자로 상부에 설치하여 수직 축에 6축 매니퓰레이터를 연결하여 검사하는 자동화된 방법이 있다. 그러나 수작업의 경우 검사자가 방사선 구역에서 작업을 해야한다는 안전상의 문제점이 있으며, 후자의 경우 대형 구조물을 격납용기 내부로 반입하기 위해 Equipment Hatch를 열어야 하며 설치에 많은 시간과 인력이 소요되므로 효율성 및 경제적인 측면에서 바람직한 방법이라고 할 수 없다.

본 연구에서는 이와 같은 육안검사를 목적으로 하는 경우 이동 및 사용이 용이한 소형 수중로봇을 이용함으로써 안전성 및 효율성을 극대화하고자 하였다. 로봇의 움직임을 정확히 제어하기 위하여 로봇의 위치와 자세추정이 필요로 하였으며, 자세추정을 위한 수단으로 비젼시스템을 이용하였다. 이에 대한 알고리즘을 제안하고 구현한다.

2. 시스템 구성 및 알고리즘

2.1 시스템 구성

경수로 원자로의 핵연료 교체시 원자로 구조물의 상단까지 냉각수를 채운후 원자로 헤드와 상부구조물 및 핵연료 집합체를 제거하면 하부구조물 내부가 드러나게 된다. 수중로봇은 포머볼트를 검사하기 위하여 그림 1과 같이 카메라가 위치한 수면으로부터 포머가 위치한 수심 약 11~15m 내에서 작업하게 된다. 이때 수중로봇의 윗면은 카메라의 축(Optical Axis)과 거의 수직인 것으로 가정한다. 또한 실제 적용될 로봇의 구조를 고려해 그림 2와 같이 로봇의 윗면에 위치추정을 위한 'ㄷ'자형으로 8개의 LED 패턴을 설치하였다. 추정되어야 할 변수들은 로봇의 위치와 방향 (x_r, y_r, θ_r)이다.

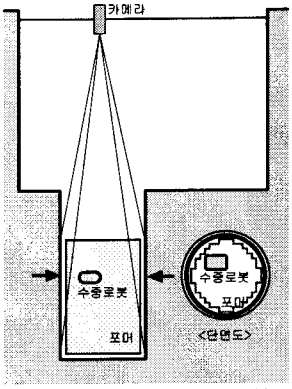

그림 1. 원자로와 수중로봇

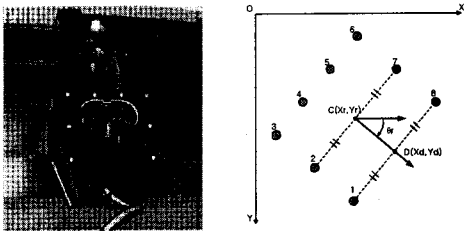

그림 2. 수중 로봇의 사진과 LED 패턴

각 LED의 번호와 위치를 찾으면 로봇의 방향과 위치를 계산할 수 있으며, 로봇의 중심위치 $C(x_r, y_r)$은 LED2와 LED7의 중점으로 정의한다. 로봇의 방향 θ_r은 점 C에서 점 D로 향하는 벡터 $\vec{v}_r$와 x축이 시계방향으로 이루는 각도이다. θ_r을 수식으로 표현하면 다음과 같다.

$$\theta_r = \tan^{-1}\left(\frac{y_d - y_r}{x_d - x_r}\right), \qquad (-\pi < \theta_r \le \pi). \qquad (1)$$

2.2 위치 추정 알고리즘

제안된 위치추정 알고리즘의 전체 구성은 그림 3과 같으며, 그림 4는 본 알고리즘을 적용해 구현한 위치 추정 프로그램의 실행 화면이다.

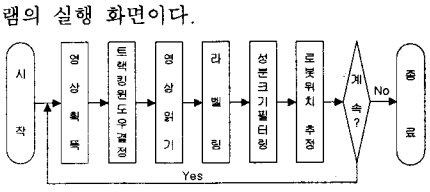

그림 3. 위치 추정 알고리즘 구성도

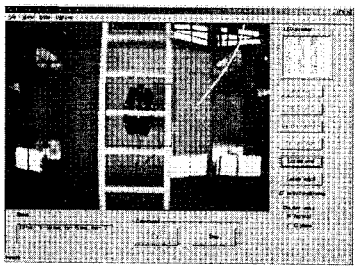

그림 4. 위치 추정 프로그램의 실행 화면

2.2.1 영상획득

전체 해상도의 영상을 이용하고 처리 속도를 높이기 위하여 동기(Synchronous)방식과 더블 버퍼링(Double Buffering)방식을 이용한다. Full Frame(640×480) 크기의 RGB 영상을 얻으며 각 성분은 {0,255} 범위의 정수값을 갖는다.

2.2.2 트랙킹 윈도우 결정

획득한 영상 중 로봇의 위치추정을 위해 처리할 영역을 결정하는 것으로 이전 프레임에서의 로봇의 위치를 이용해 트랙킹 윈도우의 위치(좌측 상단 좌표)를 결정한다.

2.2.3 영상 읽기

카메라의 감도를 고려하여 고휘도의 녹색 LED를 사용하였고, 따라서 RGB 중 G Plane에서 트랙킹 윈도우에 해당하는 영상을 읽어오며, 전체검색의 경우에는 G Plane 전체를 읽어온다.

2.2.4 라벨링

읽어온 G성분의 영상에 대해, LED에 해당하는 밝은 물체들의 위치와 크기를 구하기 위하여 임계값 처리(Thresholding)와 라벨링(Labeling)을 수행한다. 즉, 영상의 각 화소(Pixel)값을 검사하여 일정 밝기 이상인 화소들에 대한 라벨링을 수행한다. 각 물체를 구분하기 위해서는 화소들간의 연결여부를 알아야 하며, 그림 5와 같이 4-이웃화소와 8-이웃화소를 정의한다.

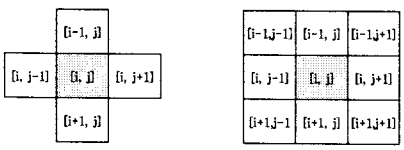

그림 5. (a)4-이웃화소 (b)8-이웃화소

본 연구에서는 여러 가지 성분 라벨링 알고리즘(Component Labeling Algorithm)들 중에서 순차 연결성분 알고리즘(Sequential Connected Components Algorithm Using 4-connectivity)을 사용하였다. 이것은 빠른 계산과 적은 메모리 사용, 영상에 대한 2번만의 조사(Scanning)로 처리할 수 있다는 장점을 갖는다.[1]

순차 연결성분 알고리즘

1) 왼쪽에서 오른쪽으로, 위에서 아래로 화상을 조사한다.
2) 만약 화소가 1값을 갖으면, 위쪽()과 왼쪽() 이웃 화소에 대하여,
   a) 두 이웃화소 중에서 하나만이 라벨을 가지고 있으면, 그 라벨을 화소에 부여한다.
   b) 두 이웃화소가 똑같은 라벨을 가지고 있다면, 그 라벨을 화소에 부여한다.
   c) 두 이웃화소가 서로 다른 라벨을 가지고 있다면, 위쪽 이웃화소의 라벨을 부여하고, 등가 테이블에 두 라벨을 등가 라벨로 기록한다.
   d) 두 이웃화소가 라벨을 갖고 있지 않다면, 새로운 라벨을 부여하고, 등가 테이블에 이 라벨을 기록한다.
3) 라벨이 부여되지 않은 화소가 남아 있으면, 다시 단계 2)로 간다.
4) 등가 테이블에서 각각의 등가 집합에 대해 가장 낮은 라벨을 찾는다.
5) 영상을 조사하여, 각 라벨을 등가 집합에서의 가장 낮은 라벨로 바꾼다.

여기서 등가 라벨은 같은 연결 성분에 대한 서로 다른 라벨을 말하며, 이 등가 라벨의 집합을 등가 집합, 그리고 등가 집합을 원소로 하는 집합을 등가 테이블이라고 한다. 이 알고리즘은 전체 화상에 대해 두 번의 조사 과정이 필요하다. 첫 번째 조사에서는 부분 영역들을 얻고 부분 영역들간의 연결성을 알아낸다(단계 1,2,3). 두 번째 조사에서는 등가인 라벨을 검사하여 다시 라벨링을 한다. 순차 연결성분 알고리즘의 예를 그림 6에 나타내었다.

0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	1	0
0	1	1	1	0	0	0	1	0	1	0
0	1	1	1	1	1	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

(a) 2진 화상(임계값 처리후의 영상)

0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	2	0
0	3	1	1	0	0	4	0	2	0	
0	3	1	1	1	1	0	4	4	2	0
0	0	0	0	0	0	0	0	0	0	0

(b) 과정 3의 결과

0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	2	0
0	1	1	1	0	0	0	2	0	2	0
0	1	1	1	1	1	0	2	2	2	0
0	0	0	0	0	0	0	0	0	0	0

(c) 과정 5의 결과

그림 6. 순차 연결성분 알고리즘의 예

2.2.5 성분 크기 필터링

라벨링 과정을 통해 얻은 성분은 LED에 의한 성분 외에 잡음이나 다른 물체의 성분을 포함할 수 있으며, 일부 LED가 수중로봇의 케이블에 의해 가려지거나 또는 작동하지 않아 해당 성분이 포함되지 않을 수도 있다. 여기서 주목할 점은, 패턴으로 사용된 8개의 LED는 유사한 성분 크기(픽셀 개수)를 이룬다. 따라서 $r_{min} \sim r_{max}$ 내의 일정한 범위의 반지름을 가지며 일정 회수 이상 나타나는 성분만을 선별하면 LED의 성분일 가능성이 크다는 점이다. 성분의 크기를 N이라 할 때, 성분의 반지름 r을 다음과 같이 정의한다.

$$r \equiv \lfloor \sqrt{\frac{N}{\pi}} \rfloor \qquad (2)$$

$\lfloor x \rfloor$는 x에 가장 가까운 정수를 의미하며, 식 (2)는 각 성분을 원형(Circle)으로 볼 때, 면적 N에 대한 반지름의 근사값을 구하는 식이다. 따라서, 성분의 반지름 r에 대한 히스토그램(Histogram)을 만들면, 잡음이 없는 이상적인 경우에는 빈도수가 8인 성분이 LED의 성분에 해당된다. 이와 같은 성분 크기 필터링을 수행하여 해당 반지름의 히스토그램 값이 임계값(보통은 4) 이상인 성분을 찾는다.

2.2.6 로봇 위치 추정

성분 크기 필터링에 의해 선별된 LED에 대응하는 성분에 대하여, 전 프레임에서의 처리 결과에 따라 결정된 검색방법에 의해 다음의 5가지 검색 방법 중 1가지가 실행된다.

1) 전체 영상에 대한 전체검색

전체 영상에서 LED 8개를 선별하여 로봇의 자세를 추정하는 방법으로서, LED 패턴을 바탕으로 2개의 LED 위치로부터 로봇의 자세를 구하기 위한 *PoseFromLedPair* 정보, 그리고 이와 반대로 로봇의 자세와 1개의 LED 위치로부터 8개의 LED 위치를 계산하는 *LedPosFromPose* 정보에 대해 설명한다.

PoseFromLedPair

*PoseFromLedPair*는 각 LED 쌍과 로봇 자세 사이의 관계를 정의한다. 가능한 쌍의 조합은 ${}_8C_2=28$ 이며, 각 조합은 방향성을 갖는 것으로 생각하여 다음과 같은 벡터들로 취급한다.

$$\vec{v}_{12},\ \vec{v}_{13},\ \vec{v}_{14},\ \ldots,\ \vec{v}_{67},\ \vec{v}_{68},\ \vec{v}_{78} \qquad (3)$$

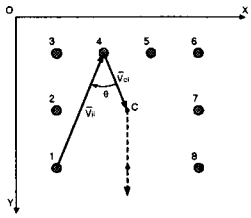

그림 7. *PoseFromLedPair*[*j*][*i*] (j=4,i=1인 경우)

그림 7과 같이 $j>i$이고 $1\le i, j\le 8$일 때, $\vec{v}_{ij}$는 LED i로부터 LED j까지의 벡터를 의미하며, LED j로부터 로봇 중심 C까지의 벡터를 $\vec{v}_{cj}$라고 하면, $\vec{v}_{ij}$가 주어질 때 *PoseFromLedPair*[*j*][*i*]는 로봇 자세를 계산하기 위한 다음과 같은 3가지 정보를 갖는다.

$$\begin{aligned}
PoseFromLedPair[j][i][0] &= \pi - \frac{\vec{v}_{cj}\cdot\vec{v}_{ij}}{\|\vec{v}_{cj}\|\,\|\vec{v}_{ij}\|},\\
PoseFromLedPair[j][i][1] &= \frac{\|\vec{v}_{cj}\|}{\|\vec{v}_{ij}\|},\\
PoseFromLedPair[j][i][2] &= \frac{\pi}{2} - \measuredangle(\vec{v}_{ij}).
\end{aligned} \qquad (4)$$

PoseFromLedPair[*j*][*i*][0]는 $\vec{v}_{ij}$와 $\vec{v}_{cj}$ 사이의 각도 θ, *PoseFromLedPair*[*j*][*i*][1]는 $\vec{v}_{ij}$와 $\vec{v}_{cj}$의 크기의 비(Scale Factor), *PoseFromLedPair*[*j*][*i*][2]는 $\vec{v}_{ij}$와 로봇 방향 사이의 각도이다.

LED*j* (x_j, y_j)와 LED*i* (x_i, y_i)가 주어지면, *PoseFromLedPair*[*j*][*i*] 정보로부터 로봇의 자세 (x_r, y_r, θ_r)를 계산할 수 있다. 로봇의 방향 θ_r은 $\vec{v}_{ij}$의 방향과 *PoseFromLedPair*[*j*][*i*][2]의 합이다. 또한 $\vec{v}_{ij}$를 LED*j*를 중심으로 *PoseFromLedPair*[*j*][*i*][0]만큼 회전시켜 점 C′의 위치를 얻으면, 로봇의 중심은 LED*j*로부터 이 점 C′으로 향하는 벡터 위에 존재한다. LED*j*와 로봇 중심 C까지의 거리는 LED*j*와 LED*i* 사이의 거리와 *PoseFromLedPair*[*j*][*i*][1]의 곱이다. 이상을 수식으로 나타내면 다음과 같다.

$$\begin{aligned}
\theta &= PoseFromLedPair[j][i][0],\\
s &= PoseFromLedPair[j][i][1],\\
x_{c'} &= (x_j-x_i)*\cos\theta-(y_j-y_i)*\sin\theta+x_j,\\
y_{c'} &= (x_j-x_i)*\sin\theta+(y_j-y_i)*\cos\theta+y_j,\\
dx &= x_{c'}-x_j,\\
dy &= y_{c'}-y_j,\\
\theta_r &= PoseFromLedPair[j][i][2]+atan2(y_j-y_i, x_j-x_i),\\
x_r &= x_j+s*dx,\\
y_r &= y_j+s*dy.
\end{aligned} \qquad (5)$$

LedPosFromPose

*LedPosFromPose*는 로봇 자세로부터 8개 LED 위치를 구하는 정보를 나타낸다. *LedPosFromPose*[*i*][0]은 LED*i*와 로봇 중심까지의 거리이며, *LedPosFromPose*[*i*][1]은 로봇 중심 C로부터 LED*i*까지의 벡터 $\vec{v}_{ic}$와 로봇 방향 사이의 각도 차, *LedPosFromPose*[*i*][2]는 LED 패턴에서 LED3과 LED4사이의 거리 d_{34} $(=d_{45}=d_{56})$이다. 이상을 수식으로 나타내면 다음과 같다.

$$\begin{aligned}
LedPosFromPose[i][0] &= \sqrt{(x_i-x_r)^2+(y_i-y_r)^2},\\
LedPosFromPose[i][1] &= atan2(y_i-y_r, x_i-x_r)-\frac{\pi}{2},\\
LecPosFromPose[i][2] &= d_{34}.
\end{aligned} \qquad (6)$$

LED*i*의 위치 (x_i, y_i)와 로봇 자세 (x_r, y_r, θ_r)이 주어지면 *LedPosFromPose* 정보를 이용해, LED*l* $(1\le l\le 8)$의 위치 (x_l, y_l)을 계산할 수 있다. 이것을 수식으로 나타내면 다음과 같다. 여기서 s는 기준 LED 패턴과 주어진 LED 패턴 사이의 크기 비이다.

$$\begin{aligned}
s &= \frac{\sqrt{(x_r-x_i)^2+(y_r-y_i)^2}}{LedPosFromPose[i][0]},\\
x_l &= s*\le dPosFromPose[l][0]\\
&\quad *\cos(LedPosFromPose[l][1]+\theta_r)+x_r,\\
y_l &= s*\le dPosFromPose[l][0]\\
&\quad *\sin(LedPosFromPose[l][1]+\theta_r)+y_r
\end{aligned} \qquad (7)$$

전체 영상에 대한 라벨링과 성분크기 필터링의 결과로 얻어진 성분들로부터 로봇 자세를 추정하는 방법을 설명한다. 제안한 알고리즘은 가정과 검증의 단계로 이루어진다. 즉 성분들의 각 쌍을 LED 쌍에 대응한다고 가정하여 식 (5)를 이용해 계산한 로봇 자세와 사용한 두 성분 중 하나를 이용해 식 (7)로부터 가상의 8개 LED 위치를 계산한다. 가상의 LED 위치와 일치하는 성분 개수(Score)를 계산하여 그 수가 8 또는 최대인 경우를 찾는다. 성분들의 가능한 쌍은 그 수가 매우 커질 수 있으며, 그림 7의 로봇 패턴에서, LED2, LED4, LED5, LED7 각각은 LED의 쌍 (1,3), (3,5), (4,6), (6,8)의 중점이라는 성질을 이용해, 검증해야할 경우의 수를 크게 줄일 수 있다.

2) 트랙킹 윈도우 내에서의 로컬검색

전체 영상이 아니라 트랙킹 윈도우에 해당하는 영상만을 처리한다. 아래에 설명할 트랙킹 윈도우 내에서의 트랙킹 방법이 실패(Fail)일 때에는, 즉 로봇의 자세를 잃어버렸을지라도 로봇은 작업 특성상 성공적으로 검출된 직전의 위치에서 크게 이동하지 않으므로, 일정 횟수만큼 재검색해 볼 필요가 있다. 검색알고리즘은 앞 절의 전체 영상에 대한 전체 검색 방법과 동일하다.

3) 트랙킹 윈도우 내에서의 트랙킹

트랙킹 방법은 전체 검색이나 로컬 검색에 의해 로봇 자세가 추정된 후부터 시작된다. 즉, LED 8개 또는 그 일부의 위치가 확인된 후에는 각 LED의 위치를 트랙킹(Feature Tracking)한다. 트랙킹의 가장 큰 장점은 트랙킹이 시작된 후, 몇 개의 LED가 꺼지거나 가려지더라도 성공적으로 로봇 자세를 추정할 수 있다는 것이다. 예를 들어, LED4와 LED5가 가려진 상태에서 전체 검색이나 로컬 검색만으로는 로봇의 8개 LED들을 선별할 수 없으나, 트랙킹 중에는 LED4와 LED5가 가려진다 하더라도 나머지 LED 6개를 선별할 수가 있다. 트랙킹을 위하여 시스템 잡음과 측정 잡음이 존재하는 환경에서도 최적의 추정을 수행하는 방법인 칼만 필터(Kalman Filter)를 이용하였다[2][3].

Kalman filter

비젼 시스템의 샘플링 시간(약 33ms)이 매우 작으므로, 로봇의 각 LED들의 움직임을 선형시스템으로 근사화할 수 있다. 따라서, 다음과 같은 선형 칼만 필터의 모델을 사용할 수 있다.

$$\begin{aligned}
\boldsymbol{x}_k &= \boldsymbol{\Phi}_{k-1}\,\boldsymbol{x}_{k-1}+\boldsymbol{w}_{k-1},\\
\boldsymbol{z}_k &= H_k\,\boldsymbol{x}_k+\boldsymbol{\mu}_k.
\end{aligned} \qquad (8)$$

여기서 $\boldsymbol{x}_k=[x_k, y_k, v_{x,k}, v_{y,k}]^T$는 상태벡터, $\boldsymbol{z}_k$는 측정값이다. $\boldsymbol{\Phi}_{k-1}$은 상태변환행렬이고, H_k는 측정행렬로서 다음과 같이 정의된다.

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (10)$$

$\boldsymbol{w}_{k-1}$은 시스템 노이즈를 모델링하고, μ_k는 측정값 노이즈를 모델링하는 zero-mean, white, Gaussian random process이다. 시스템 노이즈와 측정값 노이즈 각각의 공분산행렬(covariance matrix)들은 $Q_{k-1}=E[\boldsymbol{w}_k \boldsymbol{w}_k^T]$과 $R_k=E[\mu_k \mu_k^T]$이다. 칼만 필터 알고리즘은 상태 공분산행렬 P_k와 P_k', 이득행렬 K_k에 대한 식으로 표현된다.

$$\begin{aligned} P_k' &= \Phi_{k-1} P_{k-1} \Phi_{k-1}^T + Q_{k-1}, \\ K_k &= P_k' H_k^T (H_k P_k' H_k^T + R_k)^{-1}, \\ \hat{x}_k &= \Phi_{k-1} \hat{x}_{k-1} + K_k(z_k - H_k \Phi_{k-1} \hat{x}_{k-1}), \\ P_k &= (I - K_k H_k) P_k' (I - K_k H_k)^T + K_k R_k K_k^T. \end{aligned} \quad (11)$$

식 (9)와 (10)에서 두 행렬들이 Time-invariant이고, Q_{k-1}과 R_k을 상수행렬로 설정하면, 일정한 K_k와 P_k를 얻는다. 즉, 오프라인으로 20번 내외의 수치적인 iteration을 수행하면 Steady-State에 도달하여 일정한 K_k와 P_k을 얻는다.

본 연구에서 사용한 노이즈 공분산행렬은

$$Q_{k-1} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad R_k = \begin{bmatrix} 25 & 0 \\ 0 & 25 \end{bmatrix} \quad (12)$$

이다. 이 때 얻어진 값은 식 (13), (14)와 같다.

$$P_k = \begin{bmatrix} 15.698329 & 0.000000 & 6.099728 & 0.000000 \\ 0.000000 & 15.698329 & 0.000000 & 6.099728 \\ 6.099728 & 0.000000 & 10.294445 & 0.000000 \\ 0.000000 & 6.099728 & 0.000000 & 10.294445 \end{bmatrix} \quad (13)$$

$$K_k = \begin{bmatrix} 0.627933 & 0.000000 \\ 0.000000 & 0.627933 \\ 0.243989 & 0.000000 \\ 0.000000 & 0.243989 \end{bmatrix} \quad (14)$$

트랙킹 순서는 먼저 전 프레임에서의 각 LED의 위치와 측정된 성분들을 비교하여 서로 가장 가까이 있는 것들을 대응시킨(NNDA; Nearest Neighbor Data Association)[4] 후, 칼만 필터를 이용해 현재 프레임에서의 각 LED 위치를 추정한다. 추정된 LED 중 서로 중복되는 것은 무효화시키고, 유효 LED의 각 쌍에 대해 식 (5)와 RHT(Randomized Hough Transform)[5]을 이용해 로봇의 자세를 구한다. 계산된 로봇 자세와 식 (7)을 이용해 가상의 8개 LED 위치를 계산하고, 측정된 LED와 가상 LED가 일치하는 경우에만 LED를 최종적으로 유효화 시킨다.

4) 사용자에 의한 로봇 자세 지정

이 방법은 사용자가 LED1과 LED8은 인위적으로 마우스를 이용해 지정해 주면 식 (5)로부터 로봇 자세를 추정하고 트랙킹 방법이 실행되게 한다. 사용자에 의한 이러한 위치 지정은 알고리즘을 통해 로봇의 자세를 유일(Unique)하게 결정할 수 없을 때 유용하다. 가령, LED4와 LED5가 동시에 가려졌을 때는 알고리즘을 통해 초기의 로봇 자세로 2가지 경우의 수가 존재해 유일하게 결정할 수 없다. 이때 사용자가 LED1과 LED8을 차례로 지정해 주면, 일치하는 성분을 찾아 로봇 자세를 추정한다. 만약 LED1이나 LED8에 일치하는 성분이 없으면 사용자가 지정한 위치를 가상의 LED 위치로 간주하여 로봇 자세를 추정한다.

5) 검색 생략

사용자가 Auto-localization 기능을 실행하지 않으면, 로봇 자세를 추정하는 데 실패했을 때 프로그램이 자동으로 초기 자세를 추정하지 않고 No Operation 상태(Idle)가 된다. 이것은 LED 패턴 자체로는 유일한 자세를 결정할 수 없을 때 프로그램이 잘못된 위치를 추정하는 것을 방지한다. 즉, 잘못된 자세 추정 결과를 만들지 못하게 하는 기능으로서 안전성과 신뢰도를 높일 수 있다.

3. 실 험

본 연구에 이용한 프레임 그래이버(Frame Grabber)는 Meteor-II이며, 이것의 제어를 위해 Matrox의 MIL-Lite 6.1[3]을 사용하였다. 전체 프로그램은 Windows2000 기반의 PC(P-II 500MHz) 상에서 Visual C++6.0을 이용하여 구현하였다(그림 4 참조).

실험 결과로서, 샘플링 타임 Ts는 33ms, 초당 처리 능력은 30fps로서 NTSC에서 제공하는 최대 샘플링 비(Rate)를 제공한다. 위치 추정 알고리즘이 실행되는 시간 Tp는 7ms이다. 즉, 33-7=26ms의 여유시간이 있다. 영상 처리 후에 이 여유시간동안 로봇 제어 등의 처리를 하면, 초당 30번의 처리 능력을 갖는다. 칼만 필터를 이용한 LED들의 추적은 LED 패턴을 갑자기 빠르게 움직이지 않는 이상 잘 동작한다. 즉, Ts=33ms의 샘플링 타임에 대해, LED 패턴의 움직임이 선형 시스템으로 근사화되고 트랙킹이 잘 동작한다. 실제 적용할 수중 로봇은 관성이 크고 비교적 저속이므로 트랙킹은 강건하게 잘 동작할 것이다. 10초 동안의 트랙킹 실험에서 LED1의 위치와 속도 추정 결과를 그림 8과 그림 9에 나타내었다.

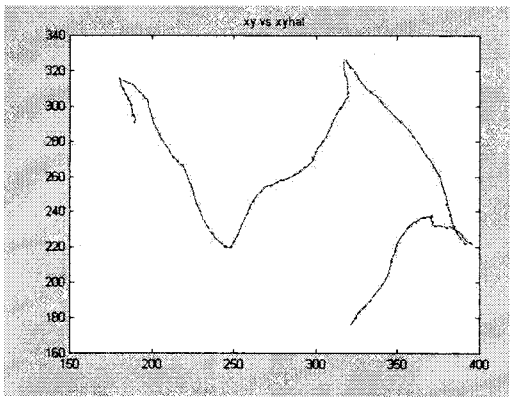

그림 8. LED1의 (x, y) 위치 추정

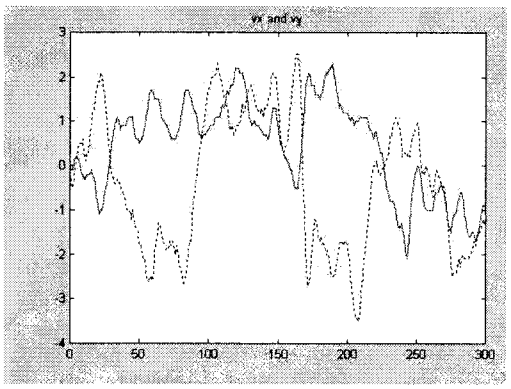

그림 9. LED1의 속도 (vx, vy) 추정

4. 결 론

본 연구에서는 수중로봇의 위치 및 자세추정을 위한 알고리즘을 제안하고 구현하였다. 프레임 그래이버를 통해 읽어들인 G성분 영상을 임계값 처리, 라벨링 및 성분 크기 필터링을 수행하여 외부 잡음이나 다른 성분들로부터 LED 패턴을 선별할 수 있었으며, 선별된 LED의 거리 및 벡터 정보를 이용함으로서 수중로봇의 위치와 방향을 계산할 수 있었다. 또한 트랙킹을 통하여 일부 LED가 가려지거나 꺼지는 경우에도 성공적으로 지속적인 로봇의 자세추정이 가능하였다.

따라서 경수로 원자로에 수중로봇과 본 알고리즘을 적용할 경우, 시스템의 소형화로 설치 및 이동이 간편하여 경비 절감의 효과가 크며, 자동화가 가능하여 방사능 구역인 작업장소를 무인화 함으로서 작업자의 방사능 노출 또한 최소화할 수 있다.

[참 고 문 헌]

[1] R. C. Jain, R. Kasturi, and B. G. Schunck, "Machine Vision", McGRAW-HILL, 1995.

[2] W. S. Cooper, "Use of Optimal Estimation Theory, in Particular the Kalman Filter, in Data Analysis and Signal Processing", Review of Scientific Instrumentation, vol. 57, pp. 2862-2869, 1986.

[3] Matrox, "MIL-Lite version 6.1 User Guide and Command Reference", March 2000.

[4] E. Trucco and A. Verri, "Instroductory Techniques for 3-D Computer Vision", Prentice-Hill, 1998.

[5] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: Randomized Hough Transform (RHT)", Pattern Recognition Letters, vol. 11, pp. 331-338, 1990