# Hungarian Algorithm을 이용한 다 개체 추적에 관한 연구

서영욱*, 이상걸*, 장원두*, 차의영*
*부산대학교 컴퓨터공학과

e-mail : fox1945@hotmail.com

# A Study on Multi-Object Tracking, Using The Hungarian Algorithm

Young-Wook Seo*, Sang-Geol Lee*, Won-Du Chang*, Eui-Young Cha*

*Dept of Computer Engineering, Pusan National University

요 약

본 논문은 여러 개체의 생물체 궤적을 효과적으로 추적하기 위해 Hungarian Algorithm을 이용한다. 생물체 궤적 정보와 생물체의 좌표 정보로 Weighted bipartite graph를 구성한다. weight는 궤적 정보와 좌표 정보의 거리, 속도, 각도를 비교하여 계산한다. 구성된 graph를 Hungarian Algorithm로 계산하여 가장 효율적인 matching이 이루어지도록 한다. 실제 생물체를 관찰하고 얻어진 데이터를 이용하여 실험을 하고, 제안한 방법의 효율성을 검증한다.

## 1. 서론

생물체의 움직임 관찰은 생물학적 연구의 기초이자 필수 작업이다. 하지만 이러한 작업은 오랜 시간, 지속적으로 관찰이 이루어 져야 하므로 많은 시간과 집중력이 필요한 일이다. 이미 이러한 작업을 인간을 대신하여 처리하는 시스템에 대한 연구가 이루어졌고, 일부 생물학적 연구에 쓰이고 있다.[1,2,3,4,5]

생물학적 연구에 있어서 다 개체 추적은 단 개체 추적보다 많은 정보를 가져다준다. 다 개체 추적은 단 개체 추적보다 같은 시간에 더욱 많은 데이터를 생성하고, 개체간에 일으키는 집단적 행동 특성을 관찰할 수 있게 한다.[4,5]

하지만 단 개체 추적에선 문제가 되지 않지만 다 개체 추적에서 해결하기 어려운 점들이 있어 개체 인식의 정확도를 떨어뜨린다. 단일 개체의 경우 실시간 영상에서 얻어지는 각 이미지마다 단일 개체의 좌표 데이터가 생성이 되고, 시간 순서에 따라 개체의 좌표를 연결하면 하나의 궤적이 나오게 된다. 다 개체 추적에선 각 개체마다 궤적이 있고, 다수 생물체 좌표 값들이 생성된다. 그리고 각 좌표가 어떤 궤적의 연장선인지를 판별해야 되는 문제가 생긴다. 앞선 연구에선 각각의 궤적과 생물체 좌표를 matching 시키는 문제를 가장 거리 변화가 적은 궤적과 좌표를 연결하는 Greedy algorithm 방식으로 풀었다.[5] 하지만 Greedy algorithm으로는 항상 최적의 해를 보장하진 않는다. 본 논문에선 이러한 궤적과 좌표 데이터간의 matching 문제를 Hungarian algorithm으로 풀어보았다. Hungarian algorithm은 weighted bipartite graph에서 최고의 weight의 합을 가지는 perfect matching을 구하여준다. 본 논문은 궤적과 생물체 좌표와의 matching ratio를 weight로 가지는 weighted bipartite graph를 구성하고 Hungarian algorithm을 적용하여 궤적, 좌표 matching 문제를 풀었다.

2장에선 앞선 연구에서 설계된 다수 물고기 추적 시스템에 대한 간략한 설명과, 추적 시스템의 다 개체 추적에서 일어나는 문제에 대해 알아보고 3장에서는 이러한 다 개체 궤적 추적에서의 문제점을 해결하기 위해 쓰인 Hungarian algorithm 에 대해 알아보겠다. 4장은 실제 생물체 추적 시스템을 통해 얻은 좌표 데이터들을 이용해 제안된 방법이 얼마만큼의 효율 증대를 보여 주는가를 확인해 보고, 5장에서 결론을 맺고, 남겨진 문제에 대해 논의하겠다.

## 2. 다 개체 추적

### 2.1 생물체 추적 시스템

다수 물고기 추적 시스템은 **그림 1**의 5 단계를 거치면서

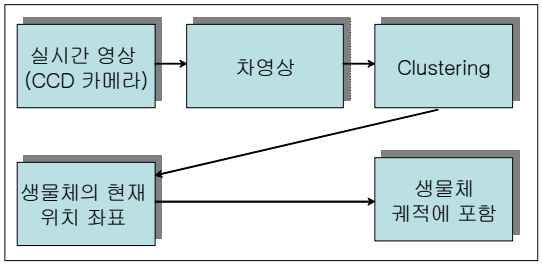


**그림 1** 생물체 추적 시스템 구성도

먼저 CCD카메라로 0.25 초마다 생물체가 활동 중인 영역을 촬영한다. **그림 2**(a)는 실제로 송사리 4개체가 있는 어항을 촬영한 장면을 보여준다. 이렇게 얻어진 영상들은 차영상을 통해 움직이는 물체 정보만 남기고, 나머지 정보는 없애 버린다. 다음, 움직임 정보를 가지고 있는 pixel들을 clustering 하여 cluster의 중심을 생물체의 중심좌표로 정한다. **그림 2**(b)는 clustering 이후 생성된 cluster의 모습이다.

(a) 실제 송사리 촬영 장면

(b) 각 송사리 위치 좌표 획득

**그림 2** 송사리 4개체의 실험 화면

이렇게 구해진 cluster들은 한 순간의 개체 위치 정보는 가지고 있어도 각 개체의 궤적 정보는 포함하고 있지 않다. 각 개체들의 궤적은 매 0.25초마다 얻어낸 cluster의 시간적 순서대로의 연결이 된다. [5]에선 궤적을 구하기 위해 각 궤적과, 각 cluster의 거리를 구한 다음 거리가 가장 짧은 궤적과, cluster를 matching하여 궤적을 구한다.

### 2. 2 다 개체 궤적 추적에서의 문제점

다 개체 궤적 추적 문제는 좌표들 중 각 궤적을 잇는 좌표가 어떤 것인지를 판단하는 문제로 요약된다.

**그림 3**은 2개체 추적 시 발생 가능한 문제를 예로 든 것이다. A와 B는 궤적의 마지막 좌표이고, X, Y는 생물체의 현재 위치 좌표라고 볼 때, A와 B에 포함될 좌표는 어떤 것인가를 판단하는 문제가 바로 궤적 추적 문제이다. A 의 경우 X, Y 둘 중 Y와 가깝고 B 역시 Y와 가깝다. 손쉬운 matching 방법으론 순차적으로 A와 Y를 연결하고, B는 나머지 X와 연결하면 된다. 그림3에서 점선으로 나타낸 matching이 이를 나타낸다. 하지만 그림에서 실선으로 보이는 matching 방법이 순차적인 방법보다 거리 차의 합이 더 짧다는 걸 알 수 있다.

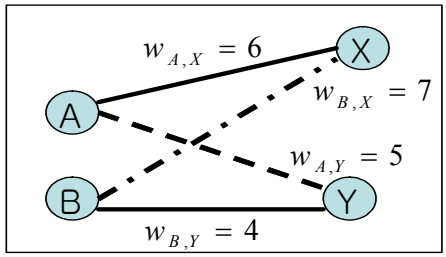


**그림 3** 2개체 궤적 추적 예

이러한 문제를 풀기 위해 가장 거리가 가까운 궤적-cluster 쌍부터 matching 시켜주는 Greedy Algorithm을 이용하였다.[5] 이 방법을 위에 제시된 예제에 적용해보면 다음과 같다. A-X, A-Y, B-X, B-Y matching 중 가장 가까운 쌍은 B-Y이므로, 가장 먼저 연결하고, 나머지 A 와 X를 연결하게 된다. Greedy Algorithm은 순차적으로 처리하는 방법보단 좋은 성능을 보이지만, 항상 최적의 matching을 보장하지는 못한다.

가장 좋은 matching을 구할 수 있는 방법은 가능한 matching을 모두 구해보면 된다. 위의 2개 개체의 경우 만들 수 있는 matching의 경우의 수가 2개이므로 2개의 matching을 직접 비교해보고 가장 좋은 matching을 택하면 된다. 하지만 이 방법은 개체 수가 n일 경우 matching을 만들 수 있는 경우의 수는 n! 가 되어 개체 수가 어느 정도 이상 늘어나면 처리 시간이 늘어나 실시간 처리는 물론, off-line에서의 데이터 처리도 어렵게 된다.

## 3. Hungarian Algorithm을 이용한 Matching

### 3.1 Hungarian Algorithm

앞서 언급한 다 개체 궤적 추적의 문제점은 Hungarian Algorithm을 적용 할 경우 매우 효과적으로 처리할 수 있다. Hungarian Algorithm 은 matching을 할 bipartite graph에서 perfect

matching이 생길 때까지 cover의 값을 점차적으로 조정함으로써 결국 최적의 matching을 찾는 방법이다. 문제가 되었던 궤적-좌표 matching 문제를 Hungarian Algorithm에 어떻게 적용하였는지를 살펴보겠다.

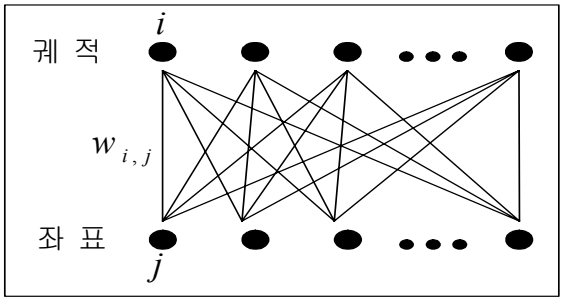


**그림 4** 궤적-좌표 Weighted Bipartite graph

**그림 4**는 궤적 데이터와 생물체 좌표와의 완전 연결을 보여주는 Bipartite graph이다. 이 graph에서 각 edge는 연결하는 궤적과 좌표간의 matching율을 가지고 있다. 각각의 edge에 가중치가 부여되고, 궤적과 좌표 vertex는 각각 하나의 좌표, 궤적 vertex와 연결이 되어야 한다. Hungarian Algorithm은 이러한 구조의 graph에서 가장 좋은 matching을 구하여준다. 먼저 각 edge가 가진 weight 정보를 모아 weight table을 구성한다.

궤적들의 집합을 $X=\{x_1,\ldots,x_n\}$,

좌표들의 집합을 $Y=\{y_1,\ldots,y_n\}$

라 했을 때 $w_{i,j}$는 $x_i$와 $y_i$를 연결하는 edge의 weight 값을 나타낸다.

$$\begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,n} \end{pmatrix}$$

행렬1. Weight table

구해진 weight table에서 cover( $u, v$) 를 $u_i = w_{i,j}$의 최대값, $v_j = 0$으로 초기화한다. 다음 $c_{i,j} = u_i + v_j - w_{i,j}$를 원소로 하는 excess matrix를 생성한다. excess matrix에서 0값을 edge로 하는 graph를 생성하였을 때의 vertex cover를 $Q$라하고, $R = Q \cap X$, $T = Q \cap Y$ 로 정한다.

초기화가 끝나면 excess matrix에서 0값을 edge로 하여 얻어진 graph에서 perfect matching이 있을 때까지 다음과 같은 수행을 반복한다.

1. $\varepsilon = \min\{u_i + v_j - w_{i,j} : x_i \in X-R, y_i \in Y-T\}$ 라 한다.
2. $x_i \in X-R$ 에 해당하는 $u_i$를 $\varepsilon$ 만큼 감소시키고, $y_i \in T$에 해당하는 $v_j$를 $\varepsilon$ 만큼 증가시킨다.
3. $c_{i,j} = u_i + v_j - w_{i,j}$ 로 excess matrix를 조정한다.

### 3. 2 Weight Table 생성

Hungarian Algorithm에 사용될 weight table은 각 궤적과 생물체 좌표 matching ratio를 객관적이고 정확히 계산되어져야 한다. 본 논문에선 궤적과 좌표간의 matching ratio를 크게 거리 차이, 속도변화, 각도 변화로 계산하였다.

거리 차이는 Euclidean distance로 구한 다음, 가능한 거리의 최댓값으로 나누어서 정규화를 하고 그 값을 $D_m$이라 정의한다.

속도변화는 궤적에 최근 1초간의 좌표 데이터의 평균 속도를 구한 다음 현재 속도와의 차이를 최대 가능 속도로 나누어서 정규화를 한다. 그 값을 $v_m$ 이라 하고 $V_m = 1 - v_m$ 라 정한다. $V_m$은 평균속도와 비슷한 궤적일 경우 1에 가까운 높은 값을, 평균속도와 차이가 나는 궤적일수록 0에 가까운 낮은 값을 가지게 된다.

각도 변화의 계산은 먼저 최근 1초간의 좌표 데이터를 통해 움직임 vector를 구한다. 구해진 vector를 통해 평균 각도 변화를 구한다. 그리고 비교되어 지는 좌표가 이루는 각도와 평균 각도의 차이를 π로 나누어 정규화를 한다. 그 값을 $a_m$이라 하고 $A_m = 1 - a_m$라 정한다.

이렇게 구해진 $D_m$, $V_m$, $A_m$은 모두 0~1 값을 가지게 되고, 0에 가까울 수록 좋지 않은 matching을, 1에 가까울 수록 좋은 matching을 뜻하는 지표가 된다. 이 3가지 지표를 이용하여 다음과 같은 weight 값을 결정하게 된다.

$$W = \alpha Dm + \beta Vm + \gamma Am \qquad (1)$$

식 (1)은 거리, 속도변화, 각도변화를 각각 가중치 $\alpha$, $\beta$, $\gamma$의 비율로 하여 weight를 결정하는 식이다. 여기서 $\alpha$, $\beta$, $\gamma$ 는 여러 번의 실험을 통해 $\alpha$ = 0.8, $\beta$ = 0.15, $\gamma$ = 0.05로 하였을 때 다른 상수 값의 실험결과 보다 더 좋은 결과를 보여주었다. 이 값은 앞으로도 더 많은 실험을 통해 세밀히 조정되어야 할 것이다.

## 4. 실험 및 결과

**그림 5는** 실험 단계를 보여주고 있다. 먼저 생물체 추적 시스템을 통해 얻은 N개의 송사리 단일 개체 좌표를 시스템의 메모리로 읽는다. 이 데이터의 첫 좌표를 각각의 궤적으로 초기화 한 다음, 두 번째 좌표부터 궤적과의 weight를 3.2장에서 설명한 방법으로 계산하여 weight table을 생성하게 한다. N*N 크기의 weight table이 생성되면 Hungarian Algorithm을 수행해 궤적과 좌표간의 matching을 한다. matching 결과에 따라 각 궤적마다 새로운 좌표를 포함시키고, 선택된 좌표가 올바른 궤적에 포함되지 않을 때마다 오류를 센다. 위 작업은 마지막 좌표 data 까지 반복 수행된다.

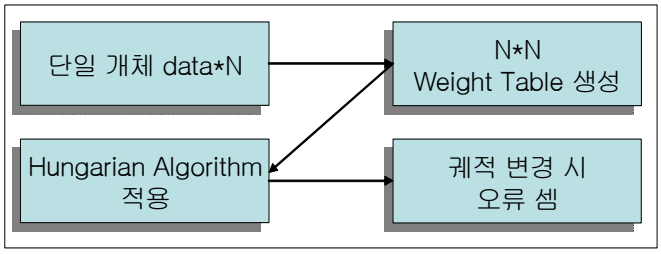


**그림 5** 실험 단계

Hungarian Algorithm 과 비교를 위해 [5]에서 쓰였던 Greedy Algorithm을 비교하였다.

data는 총 60000(250분 분량) 개의 좌표를 가진 단일 개체 data 10개를 이용하였다. 개체 수를 2~10개로 다양하게 변화를 주어 Hungarian Algorithm 과 Greedy Algorithm을 이용하여 궤적을 추적하였다. **그림6**의 그래프는 각 실험에서의 오류율을 보여 주고 있다. 그래프를 보면 개체수가 늘수록 오류율도 선형적으로 늘지만 전체적으로 Greedy Algorithm 보다 Hungarian Algorithm 기법이 더욱 좋은 결과를 보여주는 것을 확인 할 수 있었다.

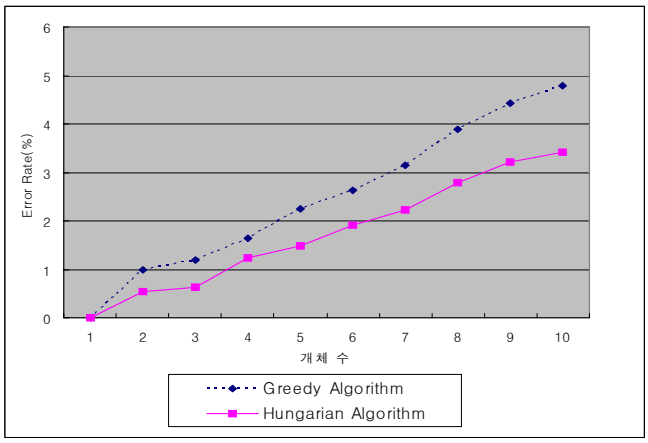


**그림 6** Greedy Algorithm과 Hungarian Algorithm 비교 실험 결과

## 5. 결론

본 논문은 다 개체 생물 움직임 데이터에서 궤적을 추적할 때 일어나는 궤적과 좌표간의 matching 문제를 Hungarian Algorithm을 통해 풀어 보았다. 실험을 통해 기존에 사용하였던 Greedy Algorithm보다 좋은 성능을 나타내는 것을 확인할 수 있었다.

하지만 기대했던 성능보단 실험 결과가 만족스럽진 않았다. 이를 개선하기 위해선 먼저 생물체 움직임을 추적하는 시스템에서 올바른 생물체의 움직임을 추적할 수 있어야 하겠다. 테스트된 데이터를 분석 결과 궤적 끊어짐 현상 등이 나타난 것이 확인되었다. 이는 올바른 궤적 추적이 어렵게 하는 요소로 작용한 것으로 생각되어 진다. 그리고 weight table 생성에서 궤적과 좌표간의 matching율을 판단하는데 있어 거리 비교, 속도 변화, 각도 변화의 가중치를 더욱 많은 실험을 통해 올바르게 조절하고, 합리적인 matching율을 계산할 수 있는 요소들을 발견해야 할 것이다.

## [참고]

[1] 김흥수, 차의영, 전태수, “배경 갱신과 반복 Clustering을 이용한 물고기 추적,” 한국 정보 처리 학회, Vol.6 No.1, pp.1375-1378, 1999

[2] 강민경, 강이철, 김성우, 차의영, "효과적인 배경 이미지를 통한 물고기 추적 기법," 한굴 멀티미디어 학회, Vol.3 No.2, pp 155~158, 2000

[3] 강동구, 차의영, 전태수, "동적인 생물체의 패턴 인식," 한국정보과학회, Vol.27 No.2, pp 437~ 439, 2000

[4] 박은경, 이상훈, 최지영, 차의영, "다중 이동 물체 추적 시의 실시간 배경 영상 갱신 방법에 관한 연구," 한국정보처리학회, Vol.10 No.1, pp 619~622, 2003.

[5] 서영욱, 차의영, “실시간 배경영상과 거리 Ranking을 통한 다개체 추적,” 한국멀티미디어학회, Vol.6 No.1 , pp575-578, 2003

[6] 강명호, 오그니언 토프로프, 차의영, "자기 조직화 신경망을 이용한 다중 표적 추적에 관한 연구, “ 한국정보과학회논문지, Vol.24, No.2 , pp525-528

[7] Douglas B. West, "Introduction to Graph Theroy," Prentice Hall, 2001