# FPGA 에 구현 가능한 KLT 추적기의 특징점 관리 방안

강우윤*, 김경환*
*서강대학교 전자공학과
e-mail : {wooyun, gkim}@sogang.ac.kr

# FPGA implementable scheme for feature points management in KLT tracker

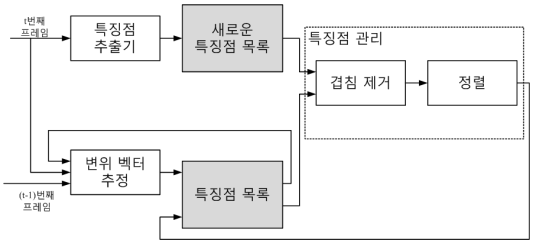
Wooyun Kang*, Gyeonghwan Kim*
*Dept. of Electronic Engineering, Sogang University

**요 약**

본 논문에서는 KLT(Kanade-Lucas-Tomasi) 추적기에서 특징점의 개수를 일정하게 유지시키기 위해 존재하는 특징점의 관리 부분을 FPGA(Field Programmable Gate Array)에 구현하기 위한 구조를 제안한다. FPGA 에 구현하기 위해 한정된 자원을 효과적으로 사용하도록 하는 것을 목표로 연산량이 많은 부분을 적은 연산량으로 구현 가능한 것으로 대체하고, 메모리의 크기와 접근 회수를 줄이기 위한 방법을 고려했다. 구현이 간단한 Harris 코너 검출기를 이용하여 특징점을 선택하고, 나눗셈 연산이 필요 없는 히스토그램을 이용하여 임계값을 설정해 특징점을 관리했다. C 언어로 시뮬레이션을 수행하여 제안한 방법을 확인했고, 기존의 특징점 관리 방법과의 비교를 통해 검증했다.

## 1. 서론

영상에서 추적이란 연속된 영상 프레임에서 점 또는 물체의 변경된 위치를 찾는 것을 의미한다. KLT 특징점 추적기는 점 기반의 추적 알고리즘으로 3D 물체 추적, 얼굴 특징 추적 등 컴퓨터 비전 분야에서 많이 사용된다[1, 2]. KLT 특징점 추적은 일반적으로 (그림 1)과 같이 진행된다[3]. (t-1)번째 프레임과 t 번째 프레임 그리고 특징점 목록에 저장되어 있는 특징점의 위치 정보를 이용하여 점들의 변위 벡터를 추정한다. 추적에 성공한 점은 특징점 목록에서 계속 유지되고, 추적에 실패한 점은 목록에서 제거한다. 추적된 특징점을 이용하여 3D 재구성을 하거나 다른 응용 분야에서 사용하기 위해서 특징점의 개수를 일정하게 유지시켜야 한다. 그러므로 추적에 실패한 점을 대신하여 t 번째 프레임에서 추출한 새로운 특징점을 추가시킨다. 이 때 추적된 점들 근처에 존재하지 않으면서, 추적에 용이게 사용되는 특징점만을 골라 목록에 추가시켜야 한다. 특징점 목록의 점의 개수를 일정하게 유지시키는 과정을 특징점 관리라고 한다.




(그림 1) 추적 시스템의 블록 다이어그램

FPGA 는 프로그래밍이 가능한 logic block 의 행렬로 이루어진 반도체 소자이다. 영상을 구성하고 있는 많은 양의 데이터를 짧은 시간 안에 연산 처리해야 할 때에는 직렬 처리를 하는 MPU(Microprocessor Unit)에서보다 병렬 처리를 할 수 있는 FPGA 에서 구현한 뒤 수행하는 것이 더 유리하기 때문에 영상처리 분야에서의 이용이 증대되고 있다[4]. 하지만 FPGA 에는 한정된 양의 logic block 에 의해 메모리 대역폭과 자원(resource) 사용의 제약이 존재한다. 그러므로 MPU 에서 수행됨을 가정한 전통적인 추적 알고리즘을 FPGA 에서 구현하고자 할 때 한정된 자원을 효과적으로 사용하기 위한 고려가 필요하다.

본 논문에서는 FPGA 의 구조상의 특성을 고려한 KLT 추적기의 구현을 위해 Harris 의 특징점 추출 방법[5]과 히스토그램을 이용한 특징점 관리 방법 및 구조를 제안하고 구현 결과를 기존의 알고리즘과 비교한다.

## 2. KLT 특징점 추적기

KLT 특징점 추적기는 $I$ 프레임의 한 점에서 인접한 $J$ 프레임의 한 점으로 이동했을 때 해당 변위 벡터를 구하는 것으로 시작된다.

$$\mathbf{d} = \arg\min_{\mathbf{d}} \int_{W} \left[ I(\mathbf{x}) - J(\mathbf{x}+\mathbf{d}) \right]^2 w(\mathbf{x}) d\mathbf{x} \qquad \text{식 1}$$

식 1 은 면적이 $W$ 인 윈도우 내에서 SSD (Sum of Squared Difference) 에러를 최소화하는 변위 벡터 $\mathbf{d}$ 를 찾는 방법을 보여준다. $w(\mathbf{x})$는 가중치 함수로 일

반적으로 윈도우 내에서 1 을 가지는 상수 함수를 선택한다. 식 1 을 불연속 함수 형식으로 변경한 뒤 1 차 Tayler expansion 을 이용하여 비선형인 부분을 선형으로 변경하여 식 2 를 얻는다[6].

$$\sum_W \left( I - J + \begin{bmatrix} I_x \\ I_y \end{bmatrix} \mathbf{d} \right) \begin{bmatrix} I_x & I_y \end{bmatrix} = 0 \qquad \text{식 2}$$

$I_x$와 $I_y$는 영상 $I$에서 $x$ 방향, $y$ 방향의 미분 값이다.

$$\mathbf{Zd} = \mathbf{e} \qquad \text{식 3}$$

$$\mathbf{Z} = \sum_W \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \ \mathbf{e} = \sum_W \begin{bmatrix} \delta I \, I_x \\ \delta I \, I_y \end{bmatrix} \qquad \text{식 4}$$

식 2 를 정리하여 식 3 과 같은 행렬식을 얻는다. 여기서 **Z** 와 **e** 는 식 4 와 같다. 이 때 **Z** 는 영상에서 크기 $W$ 인 윈도우 내부의 이차 모멘텀 행렬을 의미한다. $\delta I$ 는 영상 $I$ 와 $J$ 의 차이다. 변위 벡터 **d** 는 식 5 와 같이 행렬의 곱을 통해 구할 수 있다.

$$\mathbf{d} = \mathbf{Z}^{-1}\mathbf{e} \qquad \text{식 5}$$

영상 $I$ 에서의 특징점들로 구성된 집합에서 변위 벡터 **d** 를 이용해 영상 $J$ 로 대응되는 점을 찾는다. 대응되는 점이 정의된 영상의 경계를 벗어나거나, 행렬 **Z** 의 역행렬이 존재하지 않을 경우 특징점이 소실된 것으로 판단한다.

### 3. 기존의 특징점 관리 방법

기존의 특징점 추적 시스템에서 소실된 특징점 대신 새로운 특징점을 추가하는 과정은 다음과 같다. 모든 화소에 대해 행렬 **Z** 의 두 고유값, $\lambda_1$, $\lambda_2$ 중 코너의 강도에 해당하는 작은 고유값, $\lambda_1$ 을 구한다. 이를 내림차순으로 정렬하여 각 점의 위치 좌표와 함께 저장한다. 위치 좌표를 영상과 같은 크기를 가지는 특징점 맵에 대응시켜, 추적에 성공한 특징점과 일정 거리 내의 겹침을 확인한다. 겹침이 없는 점들을 모아 추적에 필요한 개수만을 취한다.

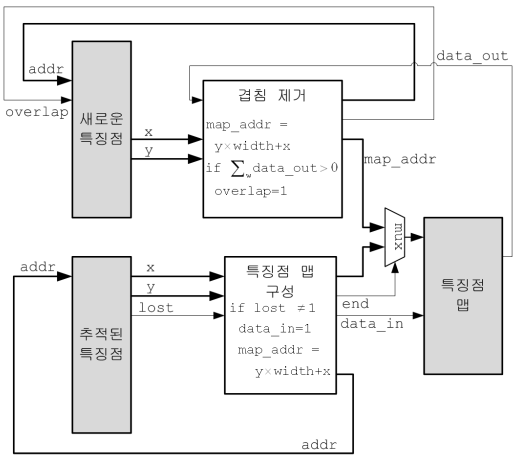
n 개의 특징점을 정렬하기 위해서 평균 $n^2$ 번의 비교를 수행해야 하고, 이와 비슷한 횟수로 메모리에 접근해야 하기 때문에 많은 시간이 걸린다. 이를 FPGA 에 구현하기 위해 히스토그램을 이용하여 코너가 가장 강한 것만을 선택할 수 있도록 한다.

### 4. 제안하는 특징점 관리 방법

기존의 방법에서 코너의 강도를 의미하는 **Z** 행렬의 고유값은 제곱근 연산을 수행해야 구할 수 있기 때문에 간단한 연산으로 표현할 수 있는 Harris 코너 추출 방식의 코너 응답으로 코너의 강도를 표시한다.

$$\begin{aligned} R &= \lambda_1\lambda_2 - k\cdot(\lambda_1+\lambda_2)^2 \\ &= Det(\mathbf{Z}) - k\cdot Trace(\mathbf{Z})^2 \end{aligned} \qquad \text{식 6}$$

식 6 에서 R 은 코너 응답을 의미하며 $k$ 는 상수이다. 모든 화소에서 특정 임계값 이상의 R 을 가지는 점을 추출한다. 그 후 NMS (Non-maximum suppression) 과정을 수행하여 지역에서 최대값을 가지는 점을 코너로 선택한다[7].




(그림 2) 특징점 사이의 겹침을 제거하는 하드웨어 구조

최종적으로 선택되는 특징점들은 일정한 거리 이상의 간격을 유지하고 있다. 또한 저장하는 코너의 개수도 기존의 방법보다 적기 때문에 사용하는 메모리의 크기도 작다. 기존의 방법 에서는 $\lambda_1$ 의 크기에 따라 정렬이 되어 있지만 제안하는 방법에서는 주사선 순서에 따라 위치 정보와 R 값이 새로운 특징점 목록에 저장되어 있다.

특징점을 관리하기 위해 겹침을 제거한 뒤 히스토그램을 통해 임계값을 설정하는 과정을 다음과 같이 수행한다.

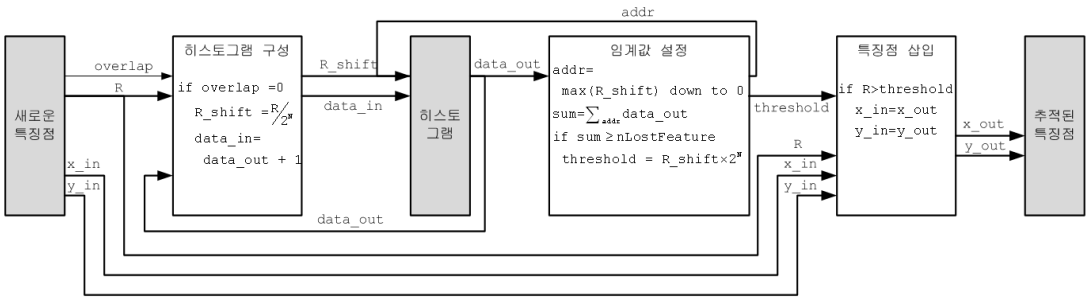
#### 4.1 특징점 사이의 겹침 제거

(그림 2)는 추적된 특징점 근처에 존재하는 새로운 특징점을 제거하는 하드웨어 구조이다. 기존의 방법과 같이 특징점 맵을 이용한다. 특징점 맵은 영상과 같은 크기의 1bit 의 데이터를 저장할 수 있는 메모리로 구성한다.

추적에 실패한 특징점 목록에는 위치 정보 `x, y` 와 함께 `lost` 를 표시하는 bit 을 1 로 표시하고 추적에 성공한 특징점에는 0 으로 표시하여 저장한다. 먼저 추적에 성공한 특징점을 특징점 맵에 표시한다. 다음으로 식 6 의 방법으로 추출한 새로운 특징점을 특징점 맵에 대응시켜 기존의 점들과의 겹침을 확인하는 순서로 수행한다.

특징점 맵에 표시하기 위해 이차원 좌표 `(x, y)` 를 일차원 좌표 `(y × width + x)`로 변환한다. 이 때 `width` 는 영상의 가로 길이이다.

겹침 제거 과정에서는 특징점 맵에서 크기 `W` 인 사각형 윈도우 내의 모든 출력 데이터(`data_out`)를 더하고, 그 결과가 0 이라면 새로운 특징점 주변에 추적에 성공한 특징점이 존재하지 않음을 의미한다. 반대의 경우에는 새로운 특징점 목록에 `overlap` bit 을 1




(그림 3) 히스토그램을 이용한 임계값 설정 과정의 하드웨어 구조

로 저장하여 다음 단계에 이 특징점을 사용하지 않도록 한다.

### 4.2 히스토그램을 이용한 임계값 설정

새로 추출된 특징점은 R 이 큰 순서대로 소실된 특징점의 개수만큼 선택되어야 한다. 히스토그램을 구성하여 임계값을 설정하고, 임계값보다 큰 값을 가지는 점을 선택하는 방법을 이용한다.

히스토그램을 구성할 때 빈(bin)의 개수가 많고, 각 빈의 범위가 작을수록 정확도는 증가한다. 하지만 전체 히스토그램을 저장하기 위한 공간도 증가하기 때문에 빈의 개수와 범위 사이의 trade-off 를 고려해서 결정해야 한다. 많은 개수의 빈을 저장하기 위해 메모리에 히스토그램을 저장한다. R 의 최소값과 최대값은 매 프레임마다 달라지는데, 저장되는 메모리의 용량을 줄이기 위해 R 의 최소값에서 최대값에 해당하는 범위를 히스토그램 빈의 전체 범위로 정한다. 각 빈의 범위를 일정한 크기로 정하기 위해서 나눗셈 연산을 수행해야 한다. 하지만 FPGA 로 구현 할 때 나눗셈 연산은 많은 지연 시간과 자원을 필요로 하기 때문에 나눗셈 연산 없이 히스토그램을 구성한다.

R 의 최소값을 $R_{min}$ 이라고 하고, N 은 $R_{min}$ 의 비트수라고 할 때 R 에 N bit 시프트 연산을 수행하여 R_shift 를 구한다.

$$R\_shift = R / 2^N \qquad \text{식 7}$$

R_shift 는 식 7 과 같은 값으로 구할 수 있다. 그리고 이를 빈의 범위로 하여 히스토그램을 구성한다. 각 빈의 값은 같은 R_shift 를 가지는 특징점의 개수이고, 빈의 간격은 R_shift 를 기준으로 1 로 정한다. 즉, 빈의 간격은 $2^N$ 이 된다. 접근이 용이하도록 히스토그램이 저장되어 있는 메모리의 주소를 R_shift 로 정한다.

히스토그램의 구성이 완료되면, 임계값을 설정한다. 원하는 개수의 특징점을 구하기 위해 R_shift 의 최대값인 max(R_shift)의 위치부터 감소하는 방향으로 빈의 값을 더한다. 누적 값이 소실된 특징점의 개수 nLostFeature 보다 커진다면 그 빈의 값을 이용하여 $R\_shift \times 2^N$ 을 수행한 뒤 이를 임계값으로 설정한다.

마지막으로 임계값보다 큰 R 을 가지는 새로운 특징점을 추적된 특징점 목록에 추가한다. 이 때 특징점의 개수가 이전 프레임의 특징점의 개수보다 많아지지 않게 하기 위해 소실된 특징점의 개수만큼을 특징점 목록에 삽입한다. 이 과정으로 특징점의 개수를 일정하게 유지시킨다. 히스토그램의 빈의 간격은 일정하지만, 나눗셈 연산을 사용하지 않고, 시프트 연산만을 이용하기 때문에 빠른 지연 시간과 적은 자원으로 원하는 임계값에 가까운 값을 얻을 수 있다. 또한 R_shift 가 메모리의 주소로 연결되기 때문에 비교기를 사용하지 않는다.

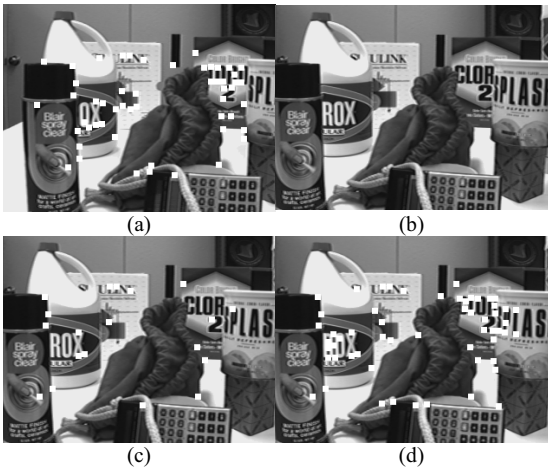
이 방법을 이용함으로써 정렬 방법에 비해 메모리에 접근하는 회수가 적기 때문에 빠른 수행이 가능하므로 FPGA 를 목적으로 하지 않는 알고리즘에도 적용이 가능하다.

## 5. 결과

제안하는 방법을 검토하기 위해 Pentium(R) D CPU 3.40GHz, Memory 2.00GB, window XP 환경의 MS Visual C++ 6.0 을 이용하여 구현했다. 비교 대상으로 open source KLT[8]를 이용했다.

(그림 4)(a)는 320×240 크기의 영상에서 추적을 수행하기 위해 추출한 특징점을 표시한 것이다. (그림 4)(b)는 5 프레임 이후의 영상이다. (그림 4)(c) 추적을 수행한 결과로 34 개의 특징점이 소실되었다. 히스토그램을 이용하여 구한 특징점의 개수는 38 개이므로 그 중 34 개만을 추가하여 (그림 4)(d)와 같은 결과를 얻었다.

1024×768 크기의 연속된 40 프레임의 영상에 대해 약 400 개의 특징점을 추적할 때 소실된 특징점은 평균 19.6 개이고, 히스토그램을 이용하여 구한 특징점의 개수는 평균 22.5 개이다. 둘 사이의 평균 오차는 3.3 개로 원하는 개수에 근접하여 추가할 특징점을 골랐음을 확인했다. 이를 통해 특징점의 개수를 일정하게 유지했다. 본 논문에서 제안하는 특징점 관리 방법을 사용하면 평균 0.016 초가 소요됐고, 퀵정렬을 이용한 open source KLT 방법을 사용하면 평균 0.188 초가 소요됐다.



(a) (b)

(c) (d)

(그림 4) 5 프레임 간격으로 추적 수행한 결과. (a)이전영상(특징점: 53 개) (b) 5 프레임 이후의 영상 (c) 추적 결과(특징점: 19 개) (d)제안하는 방법으로 새로운 특징점을 추가한 결과(특징점: 53 개)

## 6. 결론

추적 시 발생하는 특징점의 소실에 대해 새로운 특징점을 추가하는 FPGA 구조를 제안했다. Harris 코너 검출기의 결과를 NMS 과정을 사용함으로 인해 시스템 내부에 저장해야 하는 데이터의 양을 줄였고, 많은 연산량을 필요로 하는 정렬 대신 히스토그램을 이용해 원하는 개수의 특징점을 선택할 수 있다. 이 과정을 통해 메모리로의 접근 회수가 감소했고, 나눗셈 연산 대신 시프트 연산을 사용함으로써 계산 과정을 간소화 시켰다. C 언어를 이용하여 본 과정이 안정적으로 동작함을 검증했고, FPGA 에 구현 가능함을 확인했다.

## 7. Acknowledgement

본 연구는 부품소재개발사업 VISS 개발 중 ㈜만도 과제인 '반자동 주차를 위한 영상처리 하드웨어 시스템 개발' 의 지원으로 수행됨.

## 참고문헌

[1] L. Guoyuan, Z. Hongbin, H. Liu, "Affine correspondence based head pose estimation for a sequence of images by using a 3D model," Automatic Face and Gesture Recognition, 2004. Proceedings, 6th IEEE International conference on, pp. 632-637, 2004.

[2] F. Bourel, C. C. Chibelushi, A. A. Low, "Robust facial feature tracking," in Proceedings of the 11th British Machine Vision Conference, Bristol, Vol. 1, pp. 232-241, 2000.

[3] C. Tomasi, T. Kanade, "Detection and tracking of point features," Technical report, CMU-CS-91-132, Carnegie Mellon University, 1991.

[4] C. T. Johnston, K. T. Gribbon, and D. G. Bailey, "Implementing image processing algorithms on FPGAs," in Proceedings of the 11th Electronics New Zealand Conference, Palmerston North, New Zealand, pp. 118-123, 2004.

[5] C. G. Harris and M. J. Stephens, "Combined corner and edge detector," in Proceedings of the 4th Alvey Vision Conference, Manchester, pp. 147-151, 1988.

[6] S. Birchfield, "Derivation of Kanade-Lucas-Tomasi tracking equation," Unpublished, January 1997.

[7] A. Neubeck and L. V. Gool, "Efficient Non-Maximum Suppression," in Proceedings of the 18th International Conference on Pattern Recognition, IEEE Computer Society on, Vol. 3, pp. 850-855, 2006.

[8] Open source KLT : http://www.ces.clemson.edu/~stb/klt