변형된 DOT 를 이용한 클래스 검출

이희진 O, 홍기상 O

포항공과대학교 전자전기공학과 O

huijin@postech.ac.kr, hongks@postech.ac.kr

요 약

본 논문에서는 Dominant Orientation Template(DOT)을 변형하여 빠르게 같은 클래스(class)의 물체들을 찾기 위한 방법을 제안한다. 이는 같은 클래스의 DOT 들에 특정한 비트단위의 연산을 수행하여 물체들의 텍스쳐(texture)영향은 줄이면서 공통 부분은 유지하도록 하는 대표 탬플릿(template)들을 재구성한다. 따라서 DOT 알고리즘의 빠른 특성은 그대로 유지하면서 같은 클래스에 속하는 다양한 물체들을 찾을 수 있다.

1. 서론

최근 들어, 컴퓨터비전을 이용한 물체인식 및 검출 분야에 지역적 특징(local feature)을 기반으로 하는 연구[1, 2, 3, 4]가 주를 이루고 있다. 하지만, 특징점(feature-point)이 가지는 불확실성과 중의성으로 인해 이 방법들은 물체검출에 용이하지 못할 때가 있고, 또한 특징점이 거의 없는 물체의 경우에는 적용이 어려운 단점이 있다.

이를 보완하기 위해 찾고자 하는 물체를 특정 정보의 탬플릿으로 구성하여 처리영상과 비교함으로써 물체를 검출하는 탬플릿 매칭(template-matching) 방법[5]이 제안되었고, 이런 탬플릿을 구성하기 위해 가장 유용하게 쓰이는 정보들 중 하나는 그래디언트 방향성분(gradient-orientation)이다. Histograms of Gradients(HoG)[6]는 이런 방향성분들을 바탕으로 탬플릿을 구성하는 대표적인 방법이고, 이는 영상을 특정 간격으로 분할하여 하부영역(subregion)들을 얻은 후, 각 하부영역을 방향성분들의 히스토그램으로 나타낸다. 하지만, 이런 방식으로 얻어진 탬플릿은 물체의 유무를 판단하기 위해 처리 영상의 모든 영역과 비교 되어야 하고 탬플릿의 정보가 복잡할수록 실시간으로 물체를 검출하기 어렵다.

따라서 HoG 방식으로 탬플릿을 구성하는 것 대신에 각 하부영역에서 우세한 방향성분들만 선출하여 탬플릿을 만드는 Dominant Orientation Template(DOT) 방법[7]이 제안되었고, 이는 탬플릿을 처리영상과 비교 할 때 각 하부영역에서 일치하는 방향성분이 있는지 없는지만 판단하면 되므로 실시간 물체검출이 가능하다. 하지만, DOT 는 특정 물체를 기반으로 구성 되기 때문에 이에 해당하는 물체만 검출이 가능 하고<그림 1>, 같은 클래스(컵)에 속하는 다른 물체들을 검출하기 위해서는 온라인(online)상에서 그 물체에 대한 DOT 를 새로 만들어 주거나 미리 저장하고 있어야 한다. 같은 클래스에 속하는 물체들을 검출하고자 할 때, 이런 방법은 온라인 상에서의 접근이 가능하지 않거나 미리 저장해야 하는 클래스의 물체들이 많은 경우에는 적용되기 어렵다.

본 논문에서는 DOT 를 변형하여 특정 물체가 아닌 같은 클래스에 속하는 물체들을 빠르게 검출하기 위한 탬플릿 구성 방법을 제안한다.

그림 1. 왼쪽 물체의 DOT 로 매칭한 결과
(왼쪽: 찾고자 하는 물체, 오른쪽: 처리영상)

2. DOT 알고리즘

이 장에서는 본 연구의 기본이 되는 DOT 알고리즘[7]에 관해서 간략하게 설명한다.

이 알고리즘은 참조영상(reference image)의 우세한 그래디언트 방향성분들을 이용해서 DOT 를 구성하고 유사한 방식으로 구성한 입력영상(input image)에 DOT 를 매칭하여 물체를 검출한다.

참조영상을 O 라고 정의 했을 때, 이것의 DOT 를 만드는 과정은 다음과 같다.

1. 소벨 필터(sobel filter)를 사용해서 참조영상 O 의 그래디언트 정보를 얻고 이것의 방향

과 크기성분을 계산한다.

2. 참조영상 O 을 특정 간격으로 분할하여 하부영역 R 들을 만든다.
3. 각 하부영역 R 에서 크기성분을 기준으로 k 만큼의 우세한 방향성분들을 선택한다(여기서, $k=7$ 이다).
4. 참조영상 O 의 하부영역 R 에서의 우세한 방향성분들 셋(set)을 $DO(O,R)$ 라고 정의하고, 식(1)과 같이 크기성분이 기준 값보다 크면, 180° 를 정수 n_0 만큼 분할한 $[0,n_0-1]$ 의 범위 중 하나에 속하게 하고 그렇지 않으면 균일영역{⊥}으로 표현한다.

$$DO(O,R)=\begin{cases} S(O,R) \ \ if\ S(O,R)\neq\phi \\ \{\perp\} \ \ otherwise \end{cases}$$
$$with$$
$$S(O,R)=\{ori(O,l): l\in \max mag_k(R)\wedge mag(O,l)>\tau\} \quad (1)$$

여기서 $ori(O,l)$ 은 참조영상 O 의 위치 l 에서의 방향성분을 나타내고, $mag(O,l)$ 는 크기성분을 나타낸다. 또한 $\max mag_k(R)$ 는 하부영역 R 에서 k 개의 우세한 방향성분들이 선출된 위치들의 셋을 나타내며, τ 는 하부영역 R 이 균일영역인지 아닌지를 판단하기 위한 기준 값이 된다.

이와 같은 방법으로 참조영상 O 의 DOT 가 만들어지면 이를 입력영상 I 에 매칭하여 물체를 검출한다. 이때, 입력영상 I 에 대해서도 위와 유사한 과정으로 우세한 방향성분들을 얻고 여기서, $k=1$ 이다. 입력영상 I 와 이것의 한 위치 c 를 중심으로 위치한 참조영상 O 사이의 유사성(similarity) 측정은 식(2)와 같다. 이런 식(2)의 값이 가장 높은 위치 c 를 중심으로 물체가 있다고 여긴다. 여기서 $do(I,c+R)$ 은 입력영상 I 의 c 위치를 기준으로 전이된 하부영역 R 에서의 우세한 방향성분을 나타낸다. $\delta(P)$ 는 P 가 만족되면 1 을 그렇지 않으면 0 을 반환하는 함수이다.

$$\varepsilon(I,O,c)=\sum_{RinO}\delta(do(I,c+R)\in DO(O,R)) \quad (2)$$

마지막으로 식(2)의 계산 속도 향상을 위해 하부영역 R 에서 $DO(O,R)$ 와 $do(I,c+R)$ 를 8 비트로 표기한다. 즉, n_0 가 7 일 때, 처음 7 비트에 대해서는 각 방향성분이 해당하는 비트에 1 을 할당하고, 마지막 1 비트는 균일영역일 때 1 을 할당한다. 따라서 식(2)은 두 비교 영역의 비트단위 AND 연산만으로 계산이 가능해진다.

3. 제안된 알고리즘

이 장에서는 같은 클래스에 속하는 물체들을 찾기 위한 두 가지 방법으로 제시한다.

첫 번째는 가장 단순하게 DOT 알고리즘을 확장하는 방법이고, 두 번째는 본 논문에서 제안하는 것으로 DOT 를 변형하여 특정 클래스의 특성을 나타내는 대표 탬플릿들을 재구성 하는 방법이다.

3.1 DOT 알고리즘의 확장

같은 클래스에 속하는 물체들을 찾기 위한 DOT 알고리즘의 가장 단순한 확장은 해당 클래스의 물체간 변화가 어느 정도 고려 되도록 사전에 최대한 많은 물체들을 참조영상으로 가지고 DOT 를 만들어 이를 매칭할 때 이용하는 방법이다.

하지만 이 방법은 같은 클래스의 물체간 변화가 크거나 물체의 텍스쳐들이 다양한 경우에는 적용되기 어렵고, 가지고 있는 DOT 들의 수만큼 입력영상의 한 위치에서 매칭되는 수가 늘어나므로 효율적이지 않다.

3.2 제안된 방법

이 절에서는 3.1 절과 달리 사전에 구성한 많은 DOT 들에 특정 연산을 수행하여 재구성한 대표 탬플릿들을 매칭할 때 이용하는 방법을 제안한다.

<그림 2>는 컵(같은 클래스)들이 가질 수 있는 다양한 텍스쳐들을 보여준다. 본 논문은 이러한 텍스쳐(빨간 원) 영역의 영향은 줄이고 그 외에 컵의 대표가 되는 영역들(손잡이, 컵의 모양 등)은 남겨두는 클래스의 대표 탬플릿들을 구성하고자 한다.

그림 2. 컵의 다양한 텍스쳐들
(빨간 원: 텍스쳐 영역)

해당 클래스에서 이와 같은 대표 탬플릿들을 만들기 위해서는 다음과 같이 크게 두 가지 과정이 필요하다.

첫 번째는 클래스 내에서 공통적인 특성이 유사한 물체들끼리 모으기 위한 클러스터링(clustering) 과정이다. 본 논문에서는 이를 위해 상향식 클러스터링(bottom-up clustering) 방법을 사용한다.

두 번째는 특정 클러스터에서 물체들 간의 일치하지 않는 영역(텍스쳐 영역)의 영향을 줄인 대표 클러스터 탬플릿을 만드는 과정이다. 이런 방식으로 만들어진 탬플릿들은 클래스의 대표 탬플릿들을 구성할 것이다.

앞 장에서 언급 한 것처럼 한 클래스에 속하는

모든 물체영상(참조영상 O)들의 DOT 는 하부영역 R 이 8 비트로 표기된 비트들의 배열로 표현된다. 따라서, 단순한 비트단위의 연산만을 이용해 클러스터링 및 대표 클러스터 탬플릿 만드는 문제를 해결 가능하다. 다음은 이를 위한 과정을 설명한다.

1. 같은 클래스에 속하는 모든 참조영상 O 에 대해서 DOT 를 만들고 이를 T 라고 정의한다.
2. 한 클러스터를 구성하기 위해 어떤 클러스터에도 속하지 않는 탬플릿을 랜덤으로 뽑고 이를 대표 클러스터 탬플릿 C 라고 정의한다.
3. C 에서 해밍거리(hamming-distance) d_h 가 가장 가까운 탬플릿 T 을 식(3)과 같이 선택한다.

$$\arg\min_{T \notin cluster_i} \max(d_h(Cor\,T, T), d_h(C\,or\,T, C)) \quad (3)$$

여기서 or 은 비트단위 OR 연산을 의미한다.

4. 식(3)을 만족하는 d_h 값이 임계 값 τ_{dist} 보다 작으면 T 을 해당 클러스터에 넣고, 크면 과정 2 로 돌아가서 다른 클러스터를 만든다.
5. 같은 클러스터에 속하는 탬플릿들(C , T)의 하부영역을 각각 R_1 , R_2 라고 하고, 탬플릿을 구성하는 하부영역의 수를 n_{region} 이라고 하면 식(4)와 같이 비트단위 XOR(xor) 연산으로 대응하는 모든 하부영역들에 대한 비트차이를 계산할 수 있다. 비트차이가 많이 나는 영역일수록 텍스쳐 영역일 가능성이 높다.

$$diff_i = (R_{1i}\,in\,C)\,xor\,(R_{2i}\,in\,T) \quad (i = 1,...,n_{region}) \quad (4)$$

6. C 와 T 을 하나로 합쳐서 대표 클러스터 탬플릿 C 을 업데이트 할 때, 식(5)와 같이 대응하는 하부영역의 비트차이가 4 이상이면 비트단위 AND(and)연산을 이용하여 그 영역에 대한 영향을 줄이고, 그렇지 않은 영역에 대해서는 비트단위 OR 연산으로 공통적인 특성을 유지한다.

$$C(C,T,R_{1i},R_{2i}) = \begin{cases} (R_{1i}\,in\,C)\,and\,(R_{2i}\,in\,T) & if\,diff_i \geq 4 \\ (R_{1i}\,in\,C)\,or\,(R_{2i}\,in\,T) & otherwise \end{cases}$$
$$i = 1,...,n_{region} \quad (5)$$

7. 3~6 의 과정을 반복해서 한 클러스터의 대표 탬플릿 C 을 구성하고, 이때 특정 클러스터에 많은 탬플릿들이 모이는 바이어스 문제를 해결하기 위해서 최대로 묶일 수 있는 수를 임의로 정해 놓는다.
8. 2~6 의 과정을 통해 한 클래스내의 대표 클러스터 탬플릿들을 생성한다.

<그림 3>은 스테이플러와 컵 클래스에 대해서 한 클러스터에 속하는 물체들과 비트단위 AND 연산이 행해지는 영역들(빨간 사각형)을 나타낸 것이다. 이로부터 선출 된 AND 연산 영역의 대부분이 특정 텍스쳐가 속한 부분임을 알 수 있다.

(a) 스테이플러 클래스의 한 클러스터에 속하는 물체들

(b) 컵 클래스의 한 클러스터에 속하는 물체들

그림 3. 같은 클러스터에 속하는 물체들
(빨간 사각형: AND 연산 영역)

따라서 본 논문에서는 제안된 방법으로 같은 클래스 물체들의 텍스쳐 영향을 줄인 대표 클러스터 탬플릿들을 만들 수 있다. 또한 이들을 매칭할 때 이용하므로 한 영역에서의 매칭 수가 모든 DOT 의 수(3.1 절)에서 클러스터 수만큼으로 줄어든다.

4. 실험 결과 및 분석

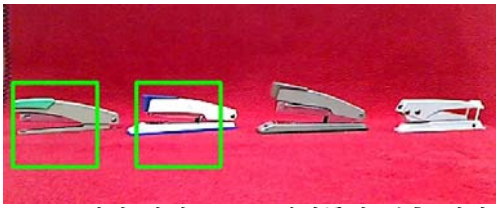
이 장에서는 3 장에서 제시한 두 가지 방법(3.1 절, 3.2 절)에 대해서 정성적인 실험과 정량적인 실험을 토대로 얻은 결과를 비교한다.

(a) 스테이플러 클래스내의 일부 영상들

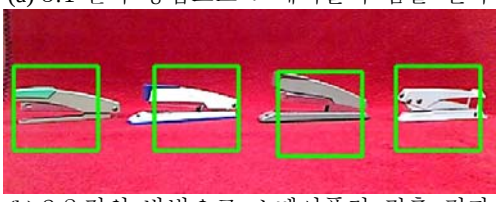
(b) 컵 클래스내의 일부 영상들

그림 4. 두 클래스에 속하는 영상들

본 논문은 두 클래스(스테이플러, 컵)를 다룬다. 스테이플러 클래스는 Caltech101[8], 구글, 네이버 등에서 모아진 총 105 장의 스테이플러 영상들로 구성되어 있으며 컵 클래스는 Caltech101,256[8,9]에서 모아진 총 142 장의 컵 영상들로 구성되어 있다. 모든 영상들은 가운데로 정렬되어 있고 영상의 사이즈는 98 x 98 로 동일하다. <그림 4>는 각 클래스 내에 구성 된 영상들의 일부를 보여준다.



(a) 3.1 절의 방법으로 스테이플러 검출 결과

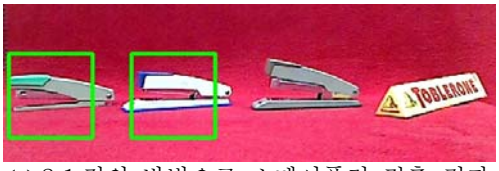


(b) 3.2 절의 방법으로 스테이플러 검출 결과

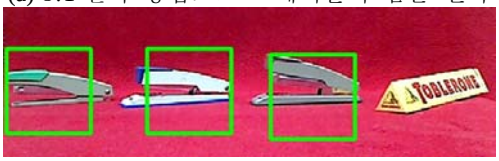
(c) 3.1 절의 방법으로 컵 검출 결과

(d) 3.2 절의 방법으로 컵 검출 결과

그림 5. 같은 클래스 속하는 물체들 검출 결과



(a) 3.1 절의 방법으로 스테이플러 검출 결과



(b) 3.2 절의 방법으로 스테이플러 검출 결과

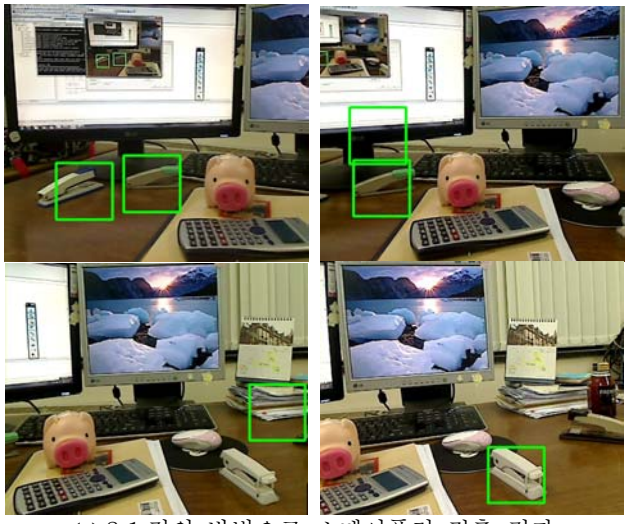
(c) 3.1 절의 방법으로 컵 검출 결과

(d) 3.2 절의 방법으로 컵 검출 결과

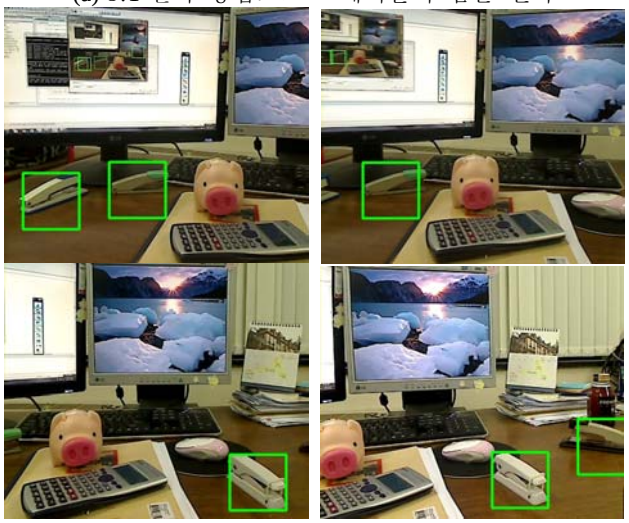
그림 6. 다른 클래스와 같이 놓았을 때 검출 결과

4.1 정성적 실험 결과

이 실험에서는 임의로 셋팅된 환경과 실제 환경에서의 스테이플러 / 컵 검출 결과 및 시간을 앞장에서 제시한 두 방법에 대해 비교하여 보여준다.



(a) 3.1 절의 방법으로 스테이플러 검출 결과



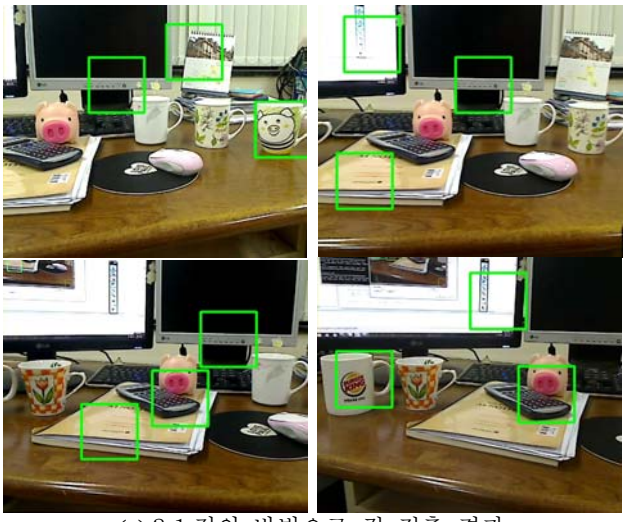
(b) 3.2 절의 방법으로 스테이플러 검출 결과

그림 7-1. 실제 환경에서 스테이플러 검출결과

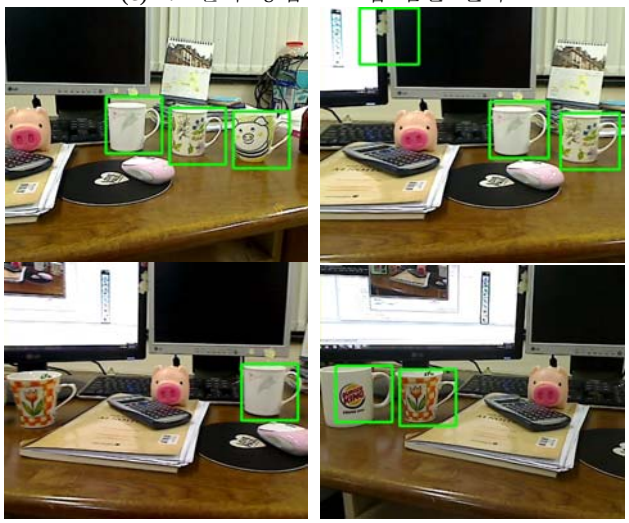
임의로 셋팅된 환경에서 같은 클래스의 물체들을 놓고 검출한 결과는 <그림 5>와 같다. 유사한 모양의 다른 클래스와 놓였을 때, 클래스 간의 구분이 얼마나 명확하게 되는지를 확인하기 위한 실험 결과는 <그림 6>과 같다. 이로부터 본 논문에서 제안된 3.2 절 방법이 3.1 절의 방법보다 같은 클래스의 속하는 물체들을 더 안정적으로 찾으면서 다른 클래스와도 명확히 구분 하는 것을 확인 할 수 있다. 또한 <그림 6-(c),(d)>로부터 3.2 절 방법이 3.1 절 방법보다 텍스쳐의 영향을 덜 받는 다는 것을 확인 할 수 있다.

실제 환경에서 스테이플러와 컵의 검출 결과는 <그림 7-1,2>와 같다. 이를 통해서 제안된 3.2 절의 방법이 실제 환경에서도 어느 정도 물체를 검출 할

수 있고, 또한 3.1 절의 방법 보다 훨씬 안정적으로 물체들을 찾는 것을 확인 할 수 있다.



(c) 3.1 절의 방법으로 컵 검출 결과



(d) 3.2 절의 방법으로 컵 검출 결과

그림 7-2. 실제 환경에서 컵 검출 결과

이 실험에서 입력영상의 사이즈는 640 x 480 으로 같고 CPU 2.67GHz, RAM 8.00GB 하드웨어 환경에서 두 방법의 테스트 시간 비교는 <표 1>과 같다. 이로부터 같은 입력 영상에 대해 3.2 절 방법이 3.1 절 방법보다 빠르게 물체를 검출 한다는 것을 확인 할 수 있고, 이는 입력영상의 한 위치에서 매칭되는 탬플릿들의 수가 대표 클러스터 탬플릿들의 수만큼으로 줄어들기 때문이다. 또한, 3.2 절 방법의 테스트 시간은 두 클래스에 대해서 각각 약 30ms 로 거의 실시간으로 동작이 가능함을 알 수 있다.

표 1. 테스트 시간 비교

	스테이플러	컵
3.1 절 방법	48ms	56ms
3.2 절 방법	31ms	33ms

4.2 정량적 실험 결과

이 실험에서는 앞 장의 두 방법에 대하여 ETHZ Shape Class Dataset[10]의 컵 클래스 검출을 수치적으로 측정하고 이를 비교 하고자 한다.

일단, 대표 클러스터 탬플릿들을 만들기 위한 참조영상 셋을 구성하기 위해서 앞서 제시된 데이터 셋의 컵 클래스에서 목적에 맞는(오른쪽 손잡이 컵) 61 장의 영상을 가지고 온다. 그리고 테스트 하기 위한 입력영상을 구성하기 위해서 ETHZ Shape Class Dataset 으로부터 컵 클래스(44), 다른 클래스(195)와 같이 총 239 장을 영상들을 가져온다. 이 입력영상에서 컵의 크기가 100width~300width 로 매우 다양하기 때문에, 실험에서 테스트 입력영상을 총 6 개의 스케일(0.8, 0.64, 0.51, 0.41, 0.33, 0.26)로 조절한다.

이 실험에서 알고리즘의 검출 정도를 측정하기 위해 사용한 방법은 0.4 FPPI (False Positive Per Image) 에서의 정확한 검출 비율이다. <표 2>는 이와 같은 방법으로 측정한 검출 정도 및 테스트 시간을 비교하여 나타낸 것이고, 이 결과로부터 제안된 3.2 절 방법이 시간으로나 성능으로나 3.1 절의 방법보다는 훨씬 좋은 것을 알 수 있다. 또한 3.2 절 방법에서의 테스트 시간이 96ms 로 늘어난 이유는 4.1 절에서의 입력영상과 달리 테스트 입력영상이 1600 x1024 만큼 큰 경우가 많아서이다. <그림 8>은 ETHZ Shape Class Dataset 의 테스트 입력영상들에 대해서 3.2 절 방법의 컵 검출 결과를 보여준다. 잘못 찾아진 경우를 살펴보면 다른 클래스의 모양이 컵의 손잡이 부분이나 컵의 옆면과 유사함을 확인 할 수 있다.

표 2. 검출 정도 측정 및 테스트 시간 비교

	Recall (%) at 0.4 FPPI (ms)
3.1 절 방법	55.74% (298ms)
3.2 절 방법	73.77% (96ms)

마지막으로 이미지 검색 비율(image retrieval rate)을 측정하여 두 방법을 비교한다. 이를 위해서 앞서 제시된 데이터 셋의 스테이플러와 컵 클래스로부터 각각 95 장의 영상과 caltech101[8]의 배경 클래스로부터 95 장의 영상을 가져와서 총 285 장의 이그잼플러(exemplar) 셋을 만들고 이그잼플러 셋에 속하지 않는 스테이플러와 컵 영상을 각각 10 장씩을 가져와서 총 20 장의 쿼리(query) 셋을 구성한다. 이와 같이 구성된 이그잼플러 셋을 앞 장에서 제시한 두 방법을 적용하여 각각 재구성하고 한 장의 쿼리영상을 넣었을 때 이로부터 쿼리영상에 해당하는 클래스를 얼마나 정확하게 검색해 내는지를 측정하여 <표 3>과 같은 결과를 얻었다. 이와 같은 결과는 더 정확한 측정을 위해 10-cross validation 을 수행하여 얻은 것이다. 이 결과로부터 제안된 3.2 절 방법이 3.1 절 방법보다 이미지 검색 시 발생할 수 있는 혼돈을 줄이는데 유용함을 알 수 있다.

(a) 잘 찾아진 경우

(b) 잘못 찾아진 경우

그림 8. ETHZ 에 대한 제안된 방법의 검출 결과

표 3. 정확한 이미지 검색 비율

	Correct retrieval rate(%)
3.1 절 방법	86.0%
3.2 절 방법	97.5%

<그림 9>는 3.1 절의 방법으로 이그잼플러 셋을 재구성 하고 이를 이미지 검색을 위해 이용할 때 잘못 검색되는 경우를 보여준다. 잘못 검색된 영상들은 컵의 특정 텍스쳐들로부터 영향을 받은 것처럼 보인다. 제안된 방법은 이러한 특정 텍스쳐로 인해 잘못 검색되는 문제를 없애기 때문에 더 정확한 검색 결과를 얻을 수 있다.

(a) 쿼리 영상들

(b) 잘못 검색된 영상들

그림 9. 3.1 절의 방법 사용시 잘못 검색된 결과

5. 결론

본 논문에서는 빠르게 특정 물체를 검출하기 위한 기존의 DOT 알고리즘[6]을 클래스를 찾기 위한 문제로 확장하기 위한 방법을 제안하였다. 이는 기존의 방법에 근거하여 탬플릿(DOT)들을 만들고, 여기에 비트단위 연산들(XOR/AND/OR)을 적용시켜 클래스의 대표 특성만 나타내는 대표 클러스터 탬플릿들을 재구성 하는 방식이다. 따라서, 제안된 방법으로 DOT 의 빠른 특성(640 x 480, 약 30ms)은 그대로 유지하면서 특정 텍스쳐들의 영향을 거의 받지 않고 같은 클래스내의 물체들을 검출 할 수 있다. 본 논문에서는 스테이플러와 컵 클래스에 대한 정성적/정량적 실험을 토대로 단순한 확장 방법과 제안된 방법을 비교하였고, 이로부터 제안된 방법이 시간적으로나 성능적으로 이보다 우세하다는 것을 확인 할 수 있다.

참고문헌

[1] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2004.

[2] H. Bay, T. Tuytelaars, and L. J. Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, 2006.

[3] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, June 2006.

[4] A. Agarwal and B. Triggs. Hyperfeatures - multilevel local coding for visual recognition. In *ECCV*, 2006.

[5] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. In *TIP*, 1997.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab. Dominant orientation templates for real-time detection of texture-less objects. In *CVPR*, 2010.

[8] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative-Model Based Vision*, 2004.

[9] G. Griffin, A. Holub, and P. Perona. Caltech 256 object category dataset. Technical Report UCB/CSD-04-1366, *California Institute of Technology*, 2007.

[10] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *ECCV*, 2006.