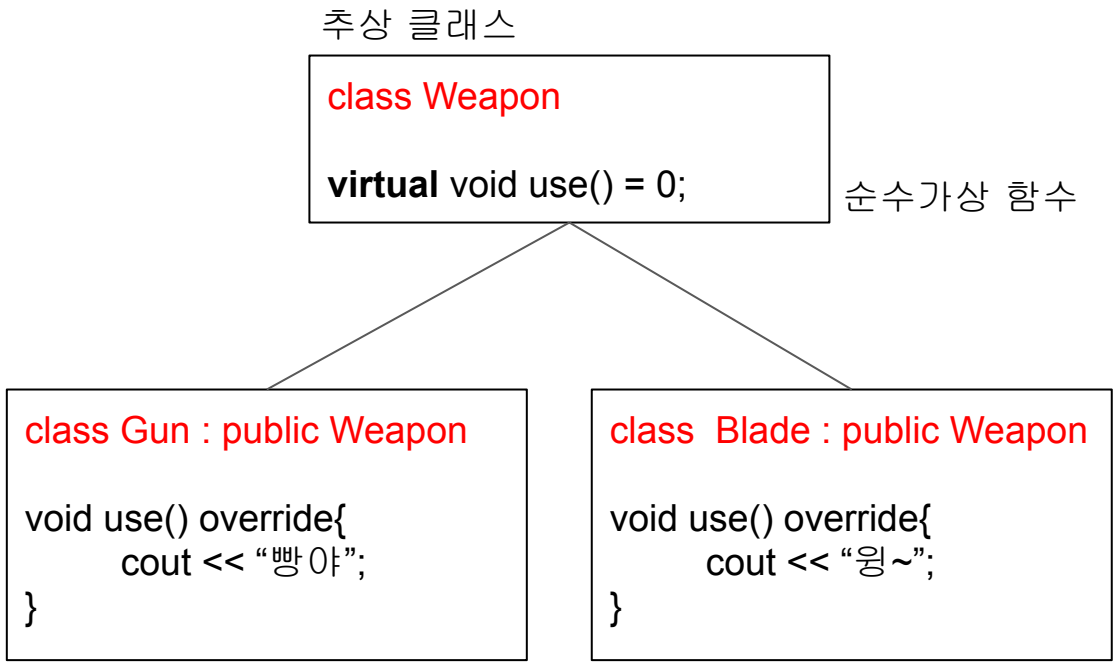
전략 패턴(Strategy Pattern)

- 널리 사용되는 디자인 패턴 중 하나
- 인터페이스와 구현을 분리함으로써 필요에 따라 구현부만 바꾸어 기능의 변경/확장을 쉽게 할 수 있게 만듬

전략 패턴(Strategy Pattern)

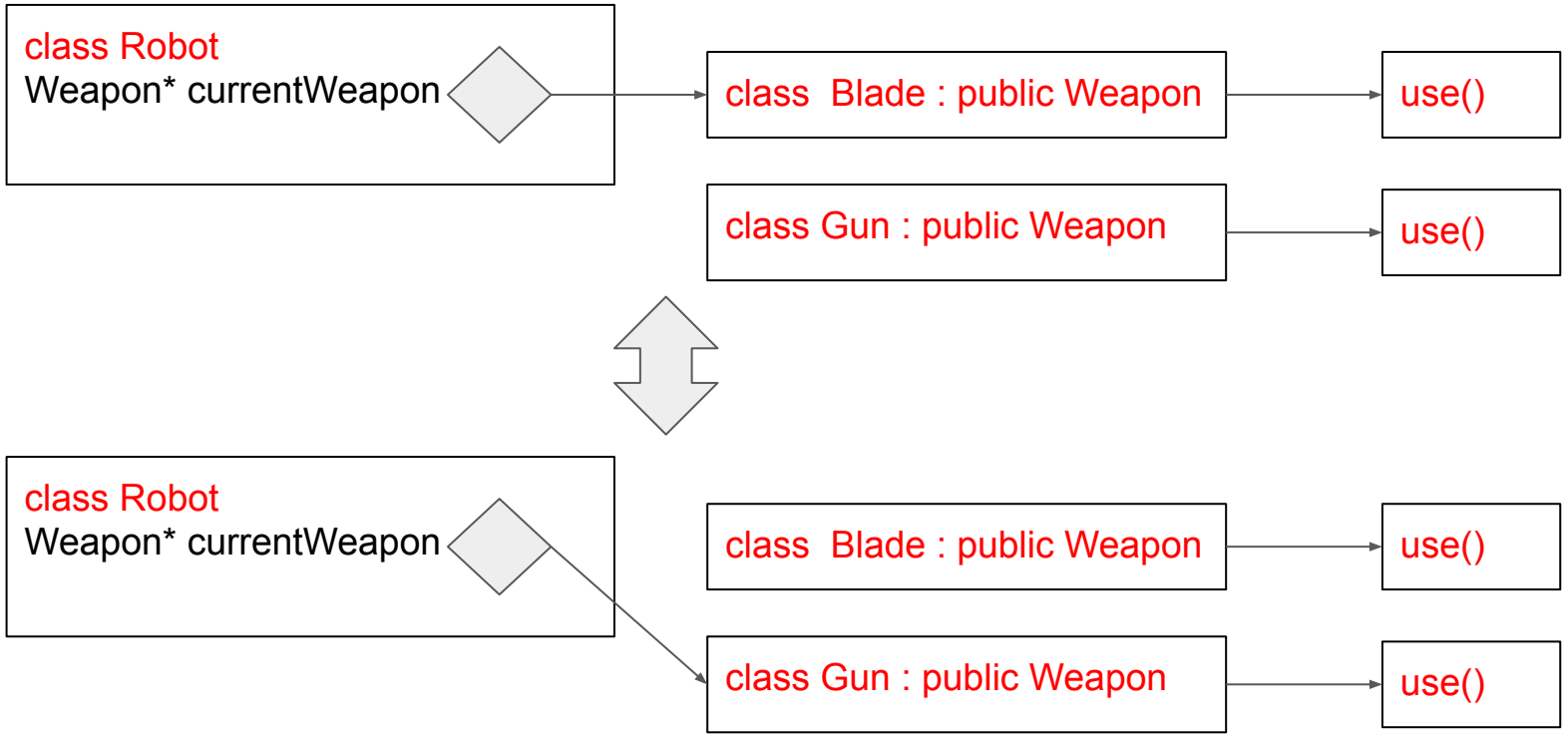
```
class Robot
private:
Weapon* currentWeapon;

public:
void setWeapon(Weapon* w){
        currentWeapon = w;
}

void attack(){

        currentWeapon->use();
}
```



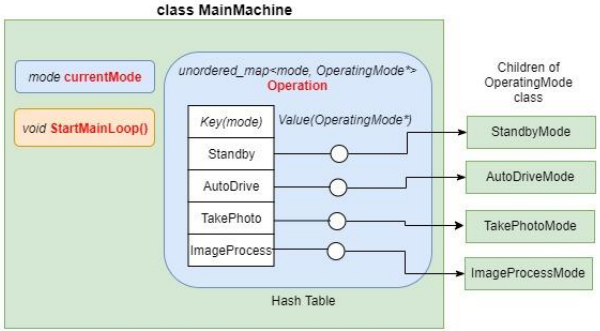
전략 패턴(Strategy Pattern)

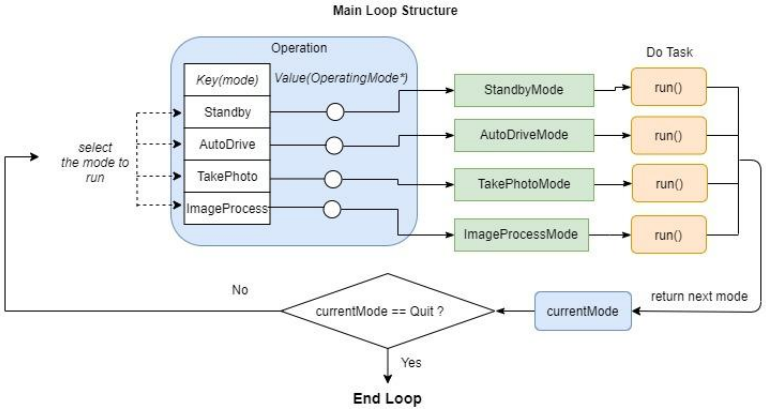
```
int main(){
        Robot robot1, robot2;   // Robot 객체 생성

        robot1.setWeapon(new Gun());
        robot2.attack();

        robot2.setWeapon(new Blade());
        robot2.attack();

        // 새로운 무기를 사용하는 새로운 로봇을 만들거나, 무기를 교체할 때
        // Weapon 클래스를 상속받는 새로운 클래스를 만들어서
        // setWeapon으로 주면 됨

        return 0;
}
```

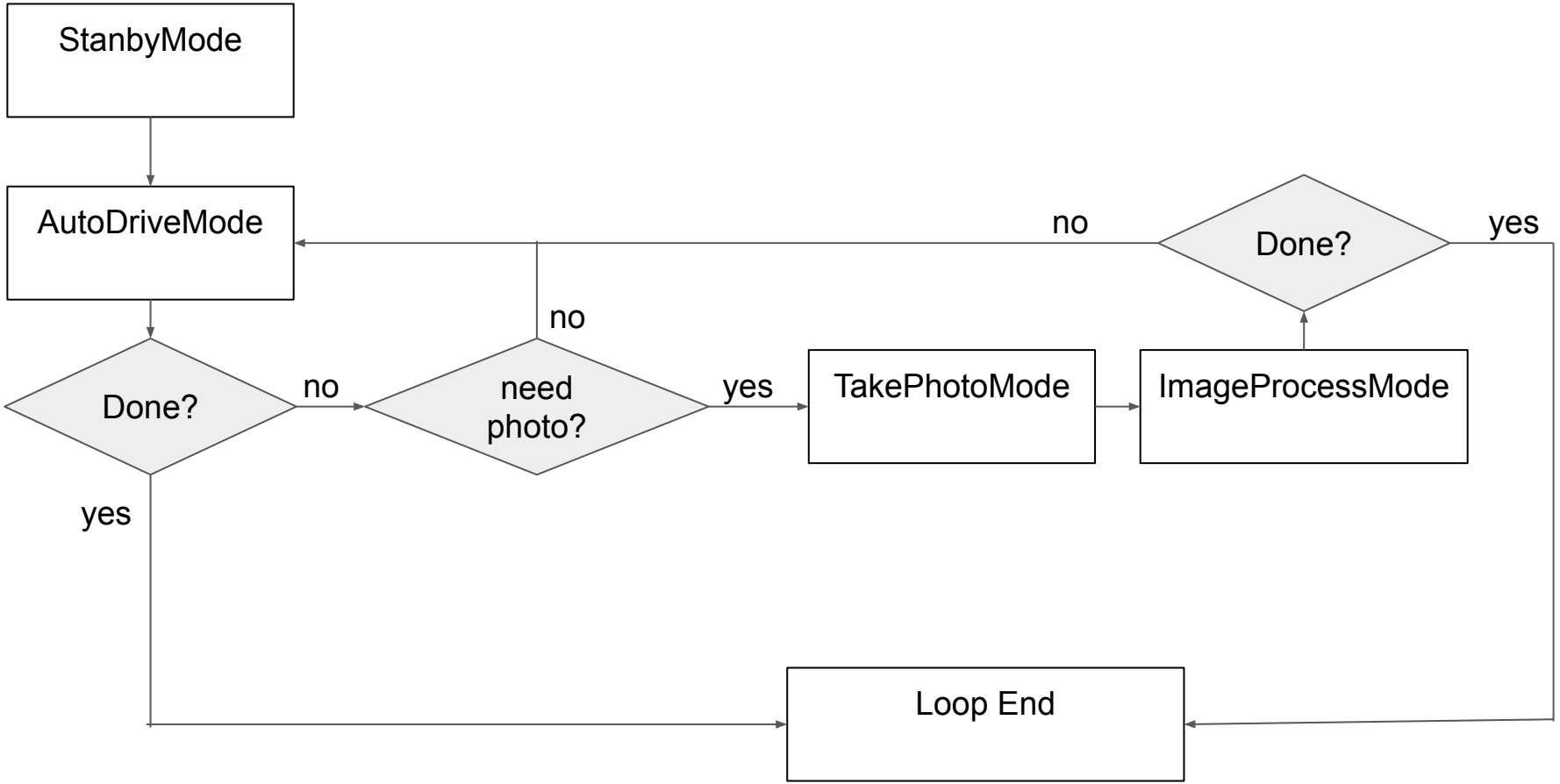
전략 패턴(Strategy Pattern)



전략 패턴(Strategy Pattern)






StanbyMode
AutoDriveMode
Done?
no
need photo?
yes
TakePhotoMode
ImageProcessMode
Done?
no
yes
yes
Loop End

MainMachine 클래스

-모드별 공통기능/자원 관리

-하드웨어 기능 대부분(이동, 사진촬영)

AutoDriveMode 클래스

-주행기능

-init 메소드에서 PathFinder 클래스에게 루트정보를 가져옴

-루트에 있는 점들을 향해 직선으로 순차적으로 이동

TakePhotoMode 클래스

-지정된 초기 각도(책장을 보는 방향)를 향하게 회전

-Lidar에서 로봇의 전방 일정 범위의 스캔정보를 가져온 후 직선검출(책장의 방향)하여 각도 보정

-Cartographer에서 맵정보 불러온 후, 로봇의 전방 일정 범위의 픽셀을 검사해서 책장의 중심위치를 계산. 책장의 중심부터 위에서 구한 방향을 기준으로 촬영위치로 이동(후진)

-위에서 계산한 책장 좌표를 바라보는 방향으로 회전

-사진촬영(1 프레임 가져옴)후 YOLO 박스 정보(책장) 가져와서 잘라냄

ImageProcessMode 클래스

-책장 사진을 가져와서
흑백화->히스토그램 평활화(명암비 증대)->엣지 검출->직선 검출

-직선중에서 수직선들을 검출하여 책들을 나눔

-직선 검출하기전에 엣지 이미지에서 짧은 선들을 제거

-너무 좁거나 수직선끼리 평행하지 않는 경우 책이 아니라고 판단

-수직선 사이(책 영역)에 엣지가 일정 픽셀 이하이면 무시

ImageProcessMode 클래스

-나눈 이미지들(책 이미지들)을 이미지 비교 알고리즘을 돌려서 무슨 책인지 찾아냄

-결과를 정리해서 Client에게 전송(문자열)

책 이미지 검색 원리

<Vocabulary Tree>

- K-Means Clustering이라는 비지도학습 러닝머신(딥러닝 아님) 알고리즘에 기반한 알고리즘

-주어진 어떤 이미지와 가장 유사한 이미지를 대규모 이미지 데이터 베이스에서 효율적으로 찾기 위한 알고리즘

Vocabulary Tree를 이해하기 전에 이해해야하는 것들

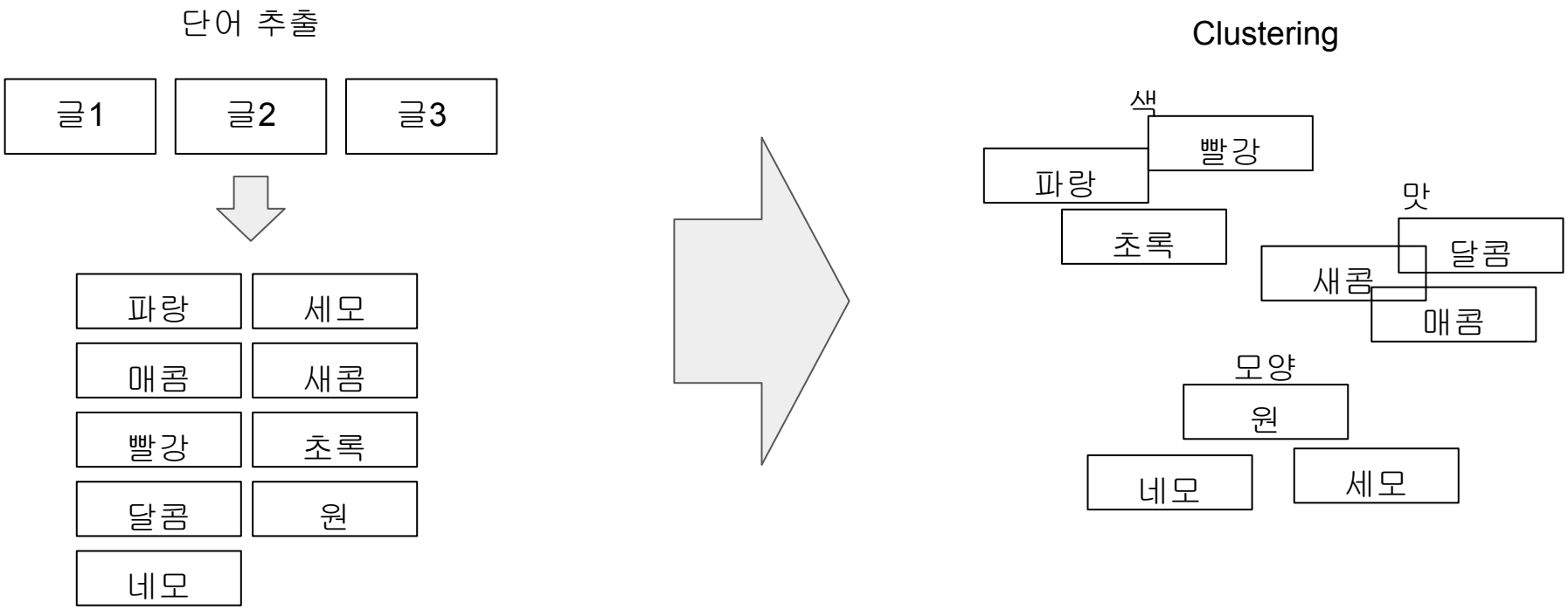
- Bag of Words(BOW)

- K-Means Clustering

- 특징점 추출 알고리즘

Bag of Words

원래는 텍스트 분석에서 사용되던 기법



Bag of Words

단어들의 등장 빈도수로 히스토그램 작성

	글1	글2	**글3**
모양에 관련된 단어	**50**	5	**49**
색에 관련된 단어	**4**	2	**7**
맛에 관련된 단어	**12**	32	**15**
...	...	...	...

글1과 글3은 단어들의 등장 빈도수 패턴이 비슷하니,
유사한 글이겠군

Bag of Words

단어들의 빈도수 => 벡터화

글1 = (50, 4, 12, ...) => (0.50, 0.04, 0.12, ...)
글2 = (5, 2, 32, ...) => (0.05, 0.02, 0.32, ...)
글3 = (49, 7, 15, ...) => (0.49, 0.07, 0.15, ...)

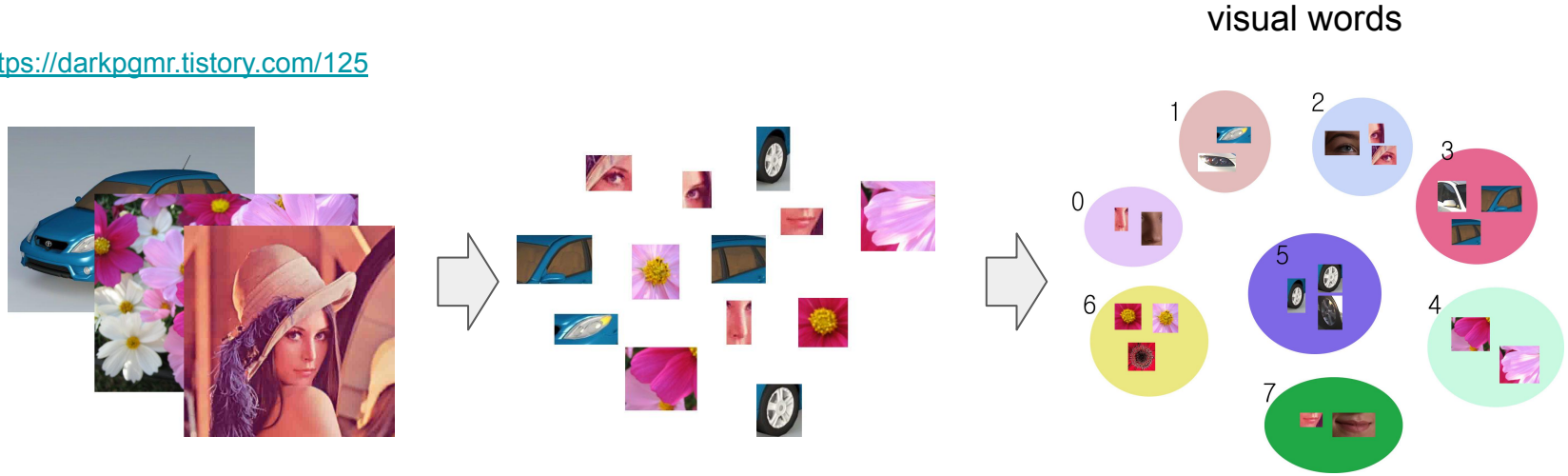
벡터간의 유사도 측정
- 거리(L1 Norm, L2 Norm), 코사인 유사도, etc...

각종 머신러닝 알고리즘의 입력값으로 활용
- SVM, etc...

Bag of Words in Computer Vision

https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

https://darkpgmr.tistory.com/125



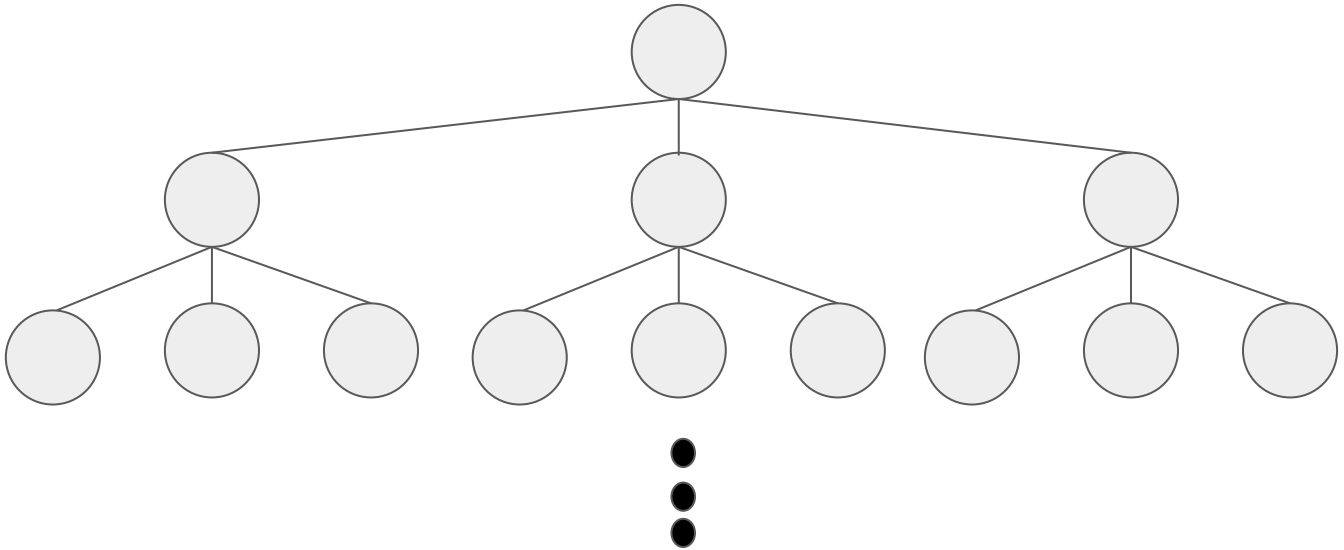
데이터 베이스
이미지1: (10, 2, 0, 3, 0, 2, ...)
이미지2: (2, 23, 2, 1, 9, 0, ...)
이미지3: (0, 0, 18, 5, 11, ...)
...

검색 이미지
(9, 3, 1, 2, 0, 2, ...)

http://www-inst.eecs.berkeley.edu/~cs294-6/fa06/papers/nister_stewenius_cvpr2006.pdf

Vocabulary Tree

- Bag of Words에서 계층적 형태(Tree)로 Clustering
- 어떤 특징 벡터가 어떤 클러스터에 속해 있는지 빠르게 검색할 수 있음
  Ex> 클러스터가 1000개라고 할때,
  일반적인 Clustering => 1000개의 클러스터와 비교
  10개씩 3층의 트리 구조로 Clustering => 30개의 클러스터와 비교



Vocabulary Tree

특징점 추출

- SIFT, SURF, KAZE, ORB 등 다양한 알고리즘이 존재함
- Scale 변화, 회전 등에 강함
- 특징점들은 주변 픽셀들의 패턴을 나타내는 다차원 벡터로 표현됨

본 프로젝트에선 특징점 추출 알고리즘으로 OpenCV Lab의 ORB알고리즘을 사용

- FAST 알고리즘 기반 (연산 속도가 빠름)
- 256 비트 이진형식의 벡터로 특징점 표현 (메모리 사용량이 적음)

Vocabulary Tree

Clustering

- K-Means Clustering 알고리즘을 이용
- 비지도 학습 머신러닝

ORB 알고리즘으로 추출한 특징점 벡터들은 이진형식의 벡터로, 고전적인 방식의 K-Means 알고리즘을 사용할 수 없음.

이진형식 벡터에서 사용할 수 있는 K-Means와 유사한 K-Majority 알고리즘 사용

http://imagelab.ing.unimore.it/imagelab/pubblicazioni/2013ElectronicImaging.pdf

Vocabulary Tree

데이터 베이스
이미지1: (0, 8, 1 ...)
이미지2: (0, 5, 4...)
이미지3: (0, 9, 0...)
이미지4: (8, 7, 0,...)
...

가중치

$$w_i = \ln (N/n_i)$$

n_i : i번째 visual word가 1번 이상 등장하는 이미지 수
N: 데이터베이스의 총 이미지 수

특정 이미지들에서만 나타나는 visual word에 높은 가중치를 곱한다

- 두 번째 word는 대부분의 이미지에서 흔하게 나타남 (덜 중요한 정보)
- 첫 번째 word는 이미지 4에서만 나타남 (중요한 정보)