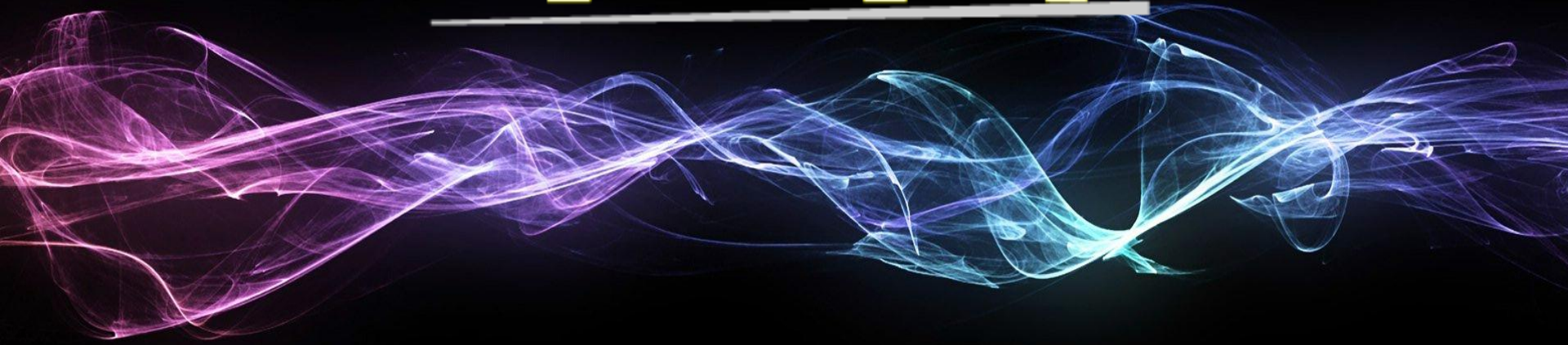
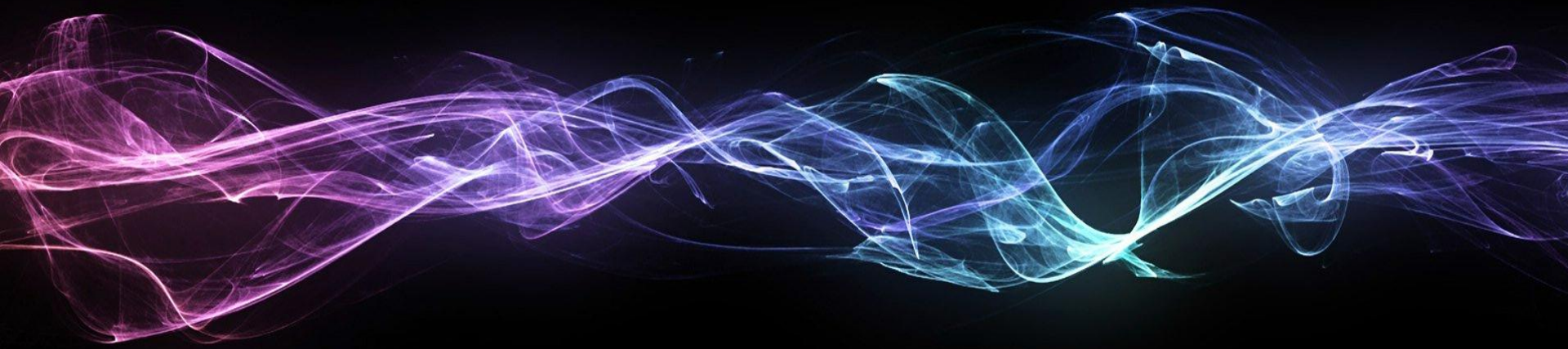
스마트폰을 이용한 간편화된 PTS(Pitch Tracking System) 어플리케이션 개발



"샤워하다생각남"
서울과학고등학교
김한성 류광민

Ⅰ. 서론



연구 동기

스포츠 분야에 응용된 정보 기술 중 상용화에 성공한 대표적 사례인
PTS (Pitch Tracking System)

- 그러나 응용 범위가 제한적이고 일반인들이 취미활동에 활용할 수 없다는 문제 발생
- 모바일 환경에 최적화된 투구 추적 기술을 개발하여 실생활에 빠르게 적용할 수 있는 방안을 모색함

기존 구속 측정 방법의 문제점

기존 구속 측정 어플리케이션*

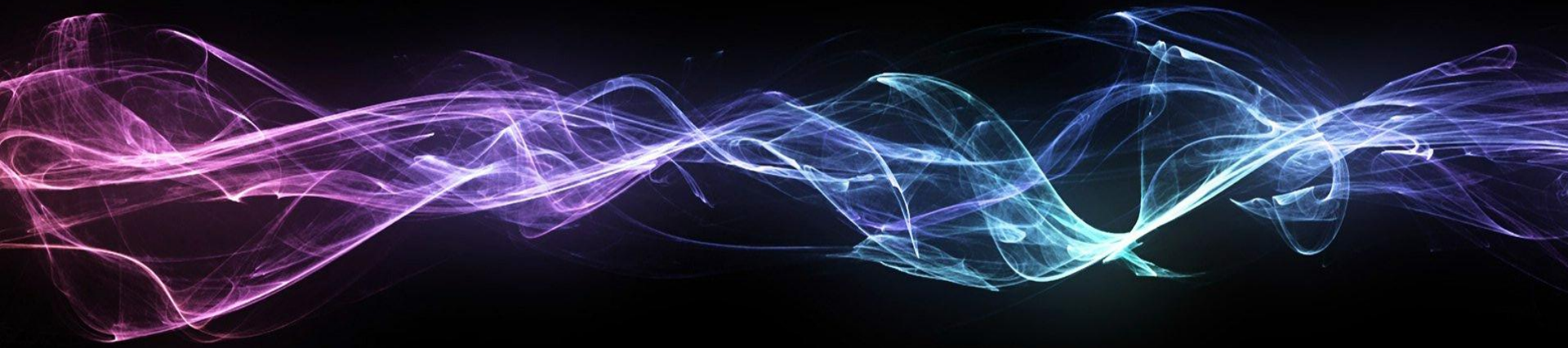
- **'Touch and Release'** – 투구의 출발, 도착 시간차를 이용한 매우 단순한 원리
- '기기를 잡고 휘두르기'
- 인간의 시각 / 반응시간에 의존해 부정확한 결과

기존 PTS (투구추적시스템)

- **고정 설치된 설비** – 휴대성 無
- 서로 다른 투구환경마다 개별적인 최적화를 거쳐야 성능을 발휘
- 매우 높은 설치 / 유지 비용

* 실제 Google play 앱 검색 결과 –
"Radar Gun", "Virtual Speed Gun", "Baseball Speed" 등

Ⅱ. 본론 - 이론 및 연구 내용



개발 기반 환경

	• Java 가상 머신에서 구동되는 Google, Inc.의 스마트폰 OS
 JavaCV	• C/C++환경에 최적화된 OpenCV와 Java간의 바인딩 라이브러리
 OpenCV	• 실시간 영상처리 라이브러리 • Open source (BSD licensed) • Cross-platform
 FFmpeg	• Open source 미디어 라이브러리 • 소리/영상 제작 및 변환에 사용

공 추적을 위한
영상 분석 알고리즘 개발 시도

Algorithm #1

HSV Color filtering (실패)

적절한 명도, 채도 제약조건(threshold)을 이용해
공이 걸러내진 흑백 이미지 생성

Algorithm #2

명도 변화 감지 (성공)

이웃한 프레임 간의 관계에서 야구공의 특성을 이용

공이 걸러내진 흑백 이미지 생성

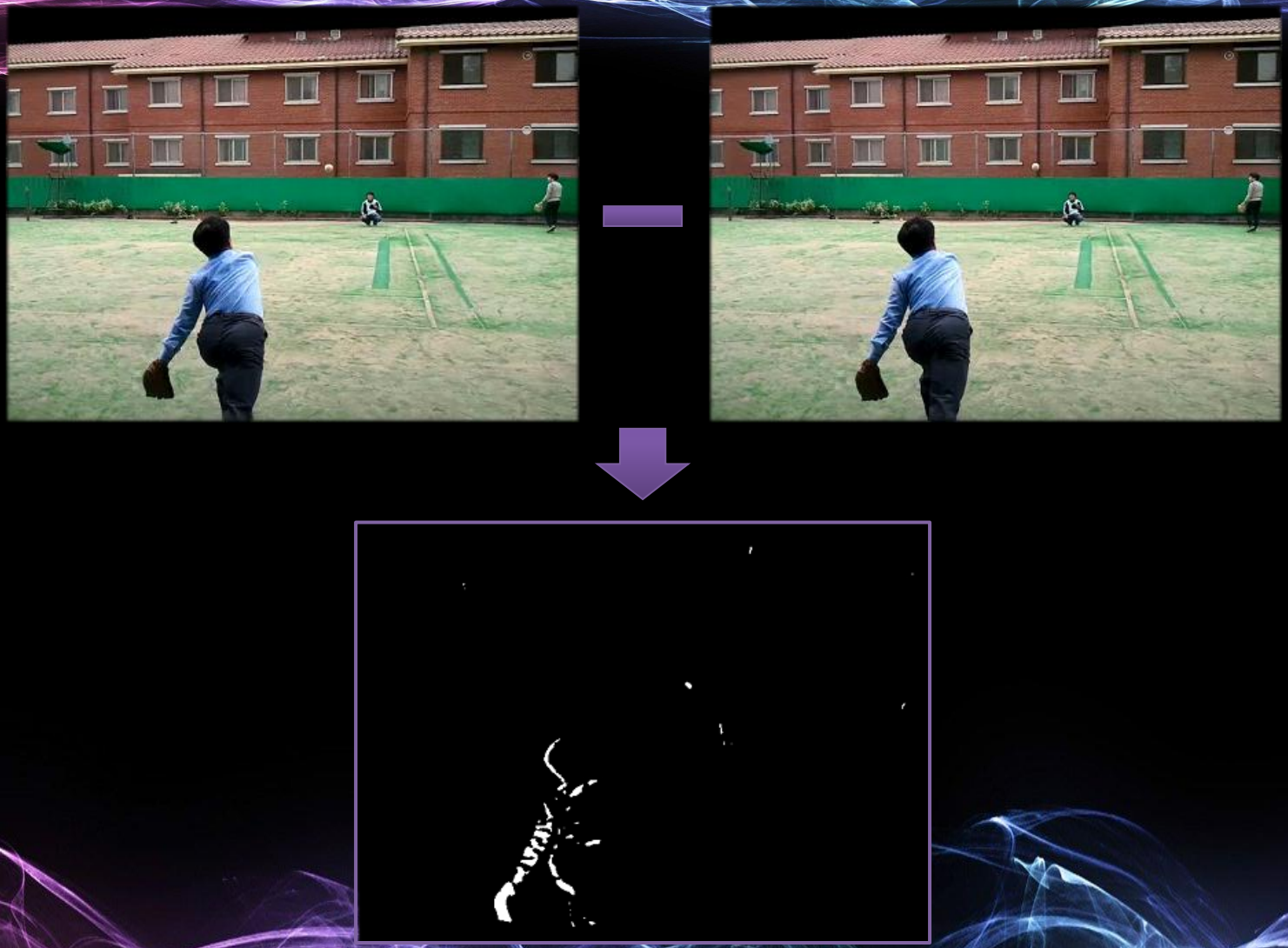
Algorithm #2의 도입

야구공만이 가지는 스패샬한 특징?

야구공은 색깔이 밝다
+ 야구공은 빠르게 이동한다 =

**야구공이 지나간 자리의 픽셀의 명도값은
다른 픽셀보다 상대적으로 그 변화가 크다**

명도 변화 감지법
Value Change Detection

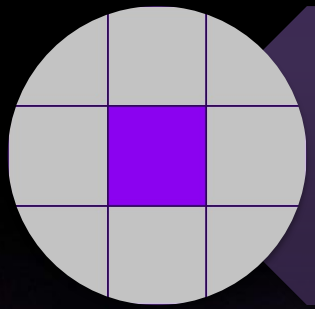


명도 변화 감지법
Value Change Detection

- 각 픽셀 하나씩 의 명도변화를 따질 경우,
  Noise, 카메라의 흔들림으로 인한 영향을 너무 많이 받는다.

Gaussian blur

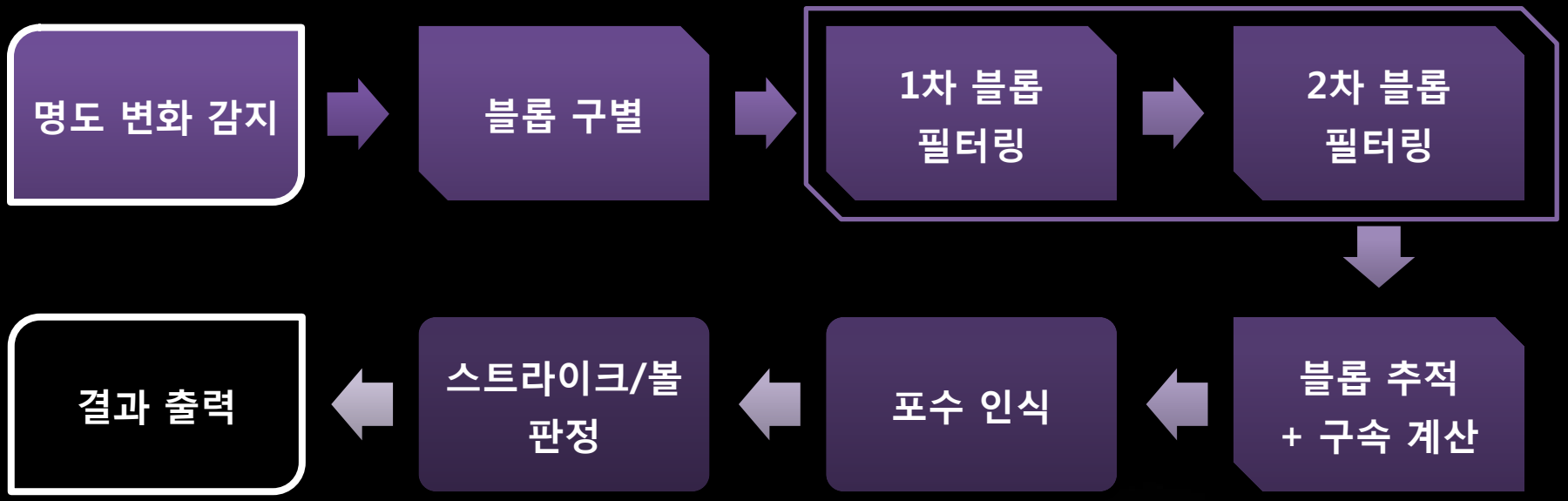
- 들쭉날쭉한 noise의 영향을 감소



각 픽셀을 중심으로 3x3 영역의 명도를 합한 값의 차이를 비교

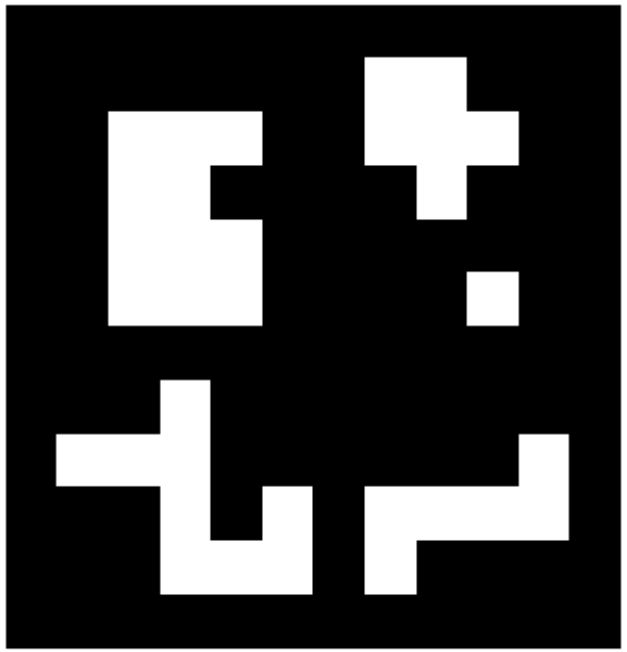
- 카메라 흔들림의 영향을 감소

전체 순서도



1. 명도 변화 감지

2. Blob Labeling



Input Image(흑백)

							1	1			
		2	2	2			1	1	1		
		2	2					1			
		2	2	2							
		2	2	2					3		
			4								
	4	4	4							5	
			4		4		5	5	5	5	
			4	4	4		5				

Blob Labeling 결과

3. Filtering Blobs

- 인식 정확도를 높이기 위해 인식 대상이 **확실히** 아닌 blob들은 최대한 제거하는 인위적 filtering 과정이 필요

Simple Blob Filtering

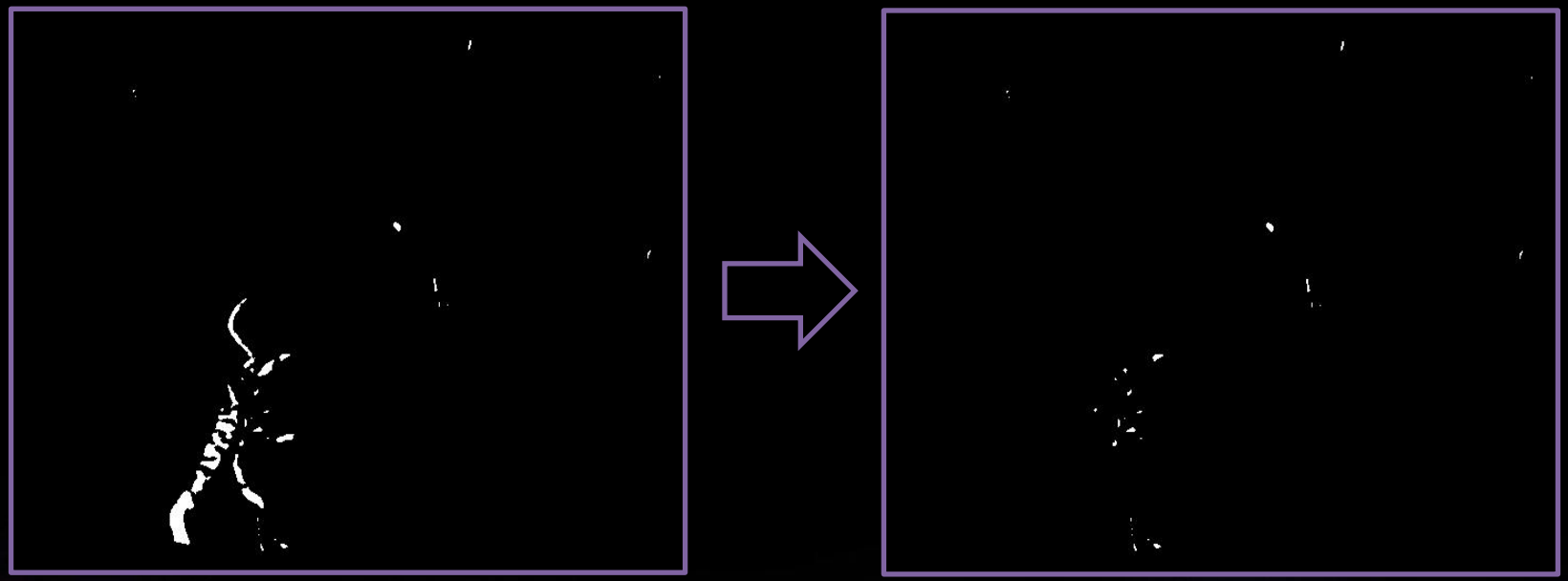
- **Blob의 크기/모양에 의존해 필터링**

Adjacent Blob Filtering

- **서로 밀집되어 있는 지역의 blob은 공이 아닐 확률이 높음**

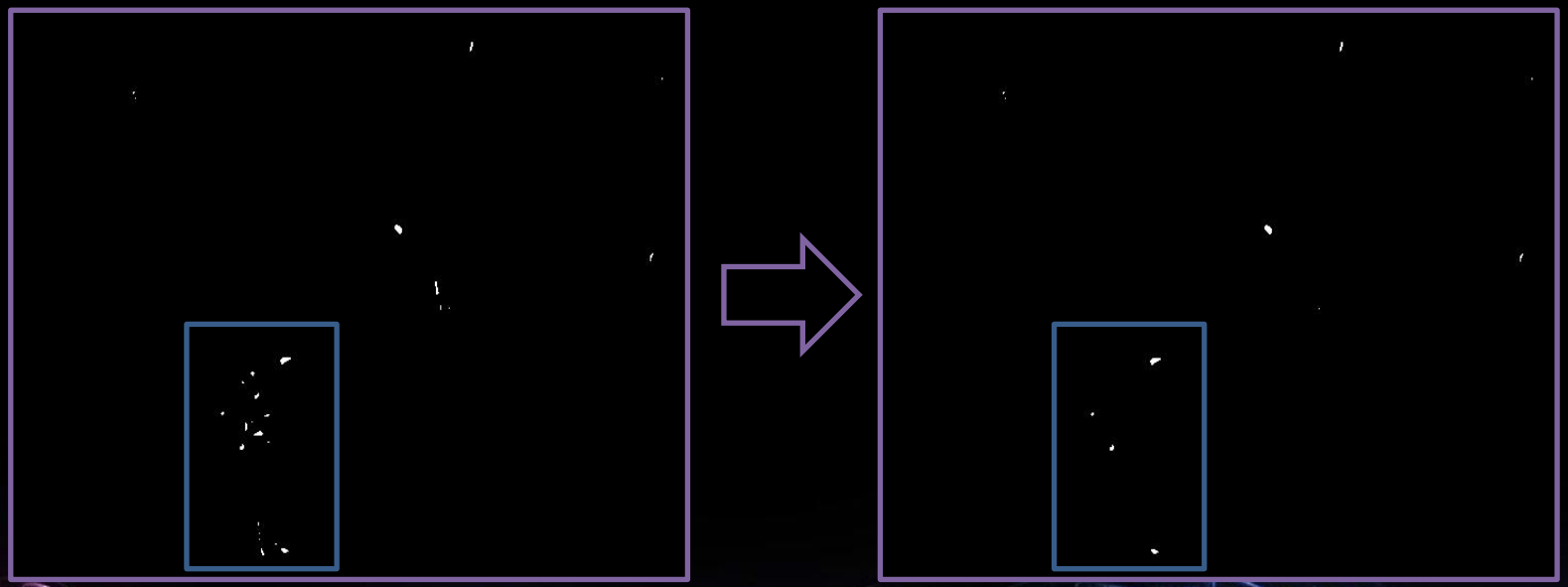
3-(1) Simple Blob Filtering

- Mass가 너무 크거나
- 가로/세로 길이 비율이 비정상적인 blob 제거



3-(2) Adjacent Blob Filtering

- 다수의 blob이 밀집되어 있는 구역은 공이 있는 부분 아님
  - 예: 투수의 몸동작 따라 생성된 '지저분한 blob'



4. Ball Tracking

- 공의 움직임을 알려면 선택된 Blob들의 frame간 운동 연관성을 밝혀야 한다.

➢ **프레임간의 여러 blob들을 어떻게 연계하여 추적하는가?**

ROI (Region Of Interest)

4. Ball Tracking

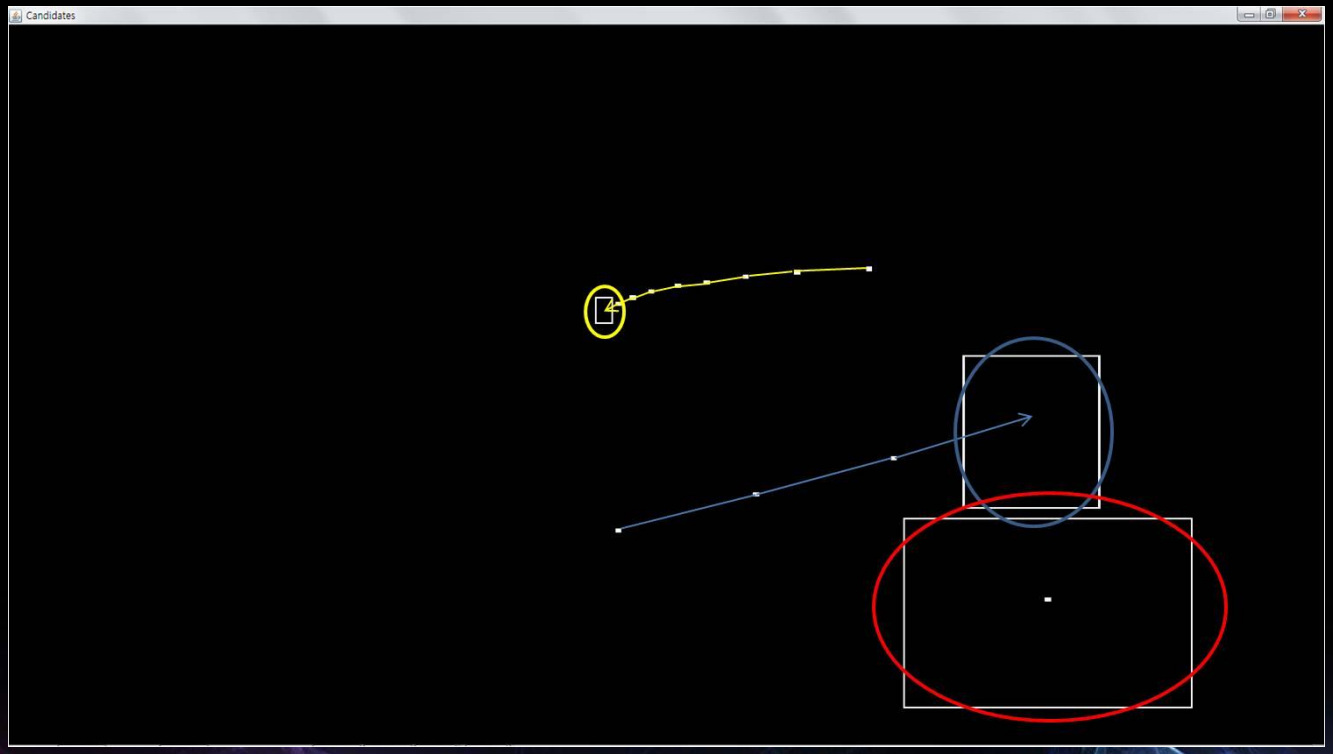
“공은 물리 법칙에 따라 운동하므로 다음 프레임에서 공의 진행 속도 및 방향은 크게 변하지 않는다.”

- 운동경로를 바탕으로 공이 탐지될 영역이 높은 관심 영역(ROI)를 선정하고 해당 영역만 탐지
- 감지 영역이 현저히 줄어들어 연산 효율 증가

ROI (Region Of Interest)

4. Ball Tracking

- 각 Candidate은 과거의 경로를 이용하여 다음에 Blob이 나타날 지점을 예측, 자동으로 ROI를 설정한다



후보(Candidate) 알고리즘

4. Ball Tracking

- ROI에서 새로운 blob이 탐지되면 Candidate라는 '보관함' 클래스에 추가됨
- Blob이 더 감지되지 않으면 Candidate는 삭제
  - **1번의 Jump 허용!**

➢ Candidate가 충분히 길어지면
다른 Candidate를 모두 kill,
그 Candidate을 공으로 간주, 사라질 때 까지 추적

4. Ball Tracking

5. 구속 계산

- 변수

  F: 공이 비행한 프레임 수 (=최종 Candidate의 길이)

  f: 영상의 frame rate (Hz)

  L: 투수와 포수 간 직선거리

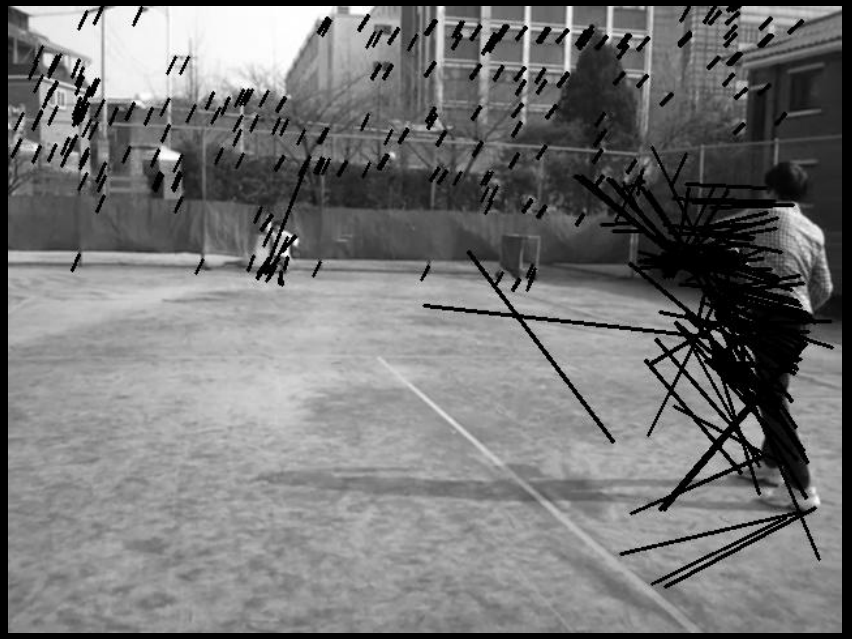
$$v = \frac{L}{F \cdot \left(\frac{1}{f}\right)} \text{ (m/s)} = \frac{3.6L \cdot f}{F} \text{ (km/h)}$$

6. 포수 인식

- Canny edge detection
- 공이 최종으로 인식된 위치 주변에서 포수 모양의 윤곽을 감지

손떨림 보정 : Optical Flow

6. 포수 인식

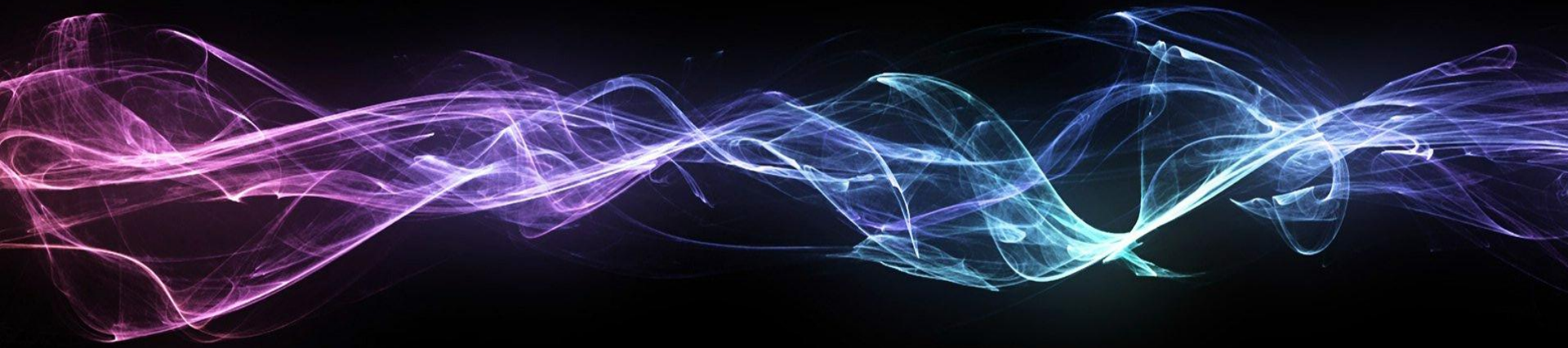


- OpenCV의 Optical Flow를 이용해 부차적인 손떨림 보정 수행
- 그러나 효과 대비 성능 희생이 커서 **최종 버전에서 탈락**

7. 스트라이크/볼 판정

- 포수의 위치를 알면 Strike zone의 위치도 파악된다
- (너비/높이는 포수-투수 원근법에 맞게 조정)
- 간단한 불 연산으로 공이 zone 안에 있는지 판정

IV. 본론 – 한눈에 보는 알고리즘 진행

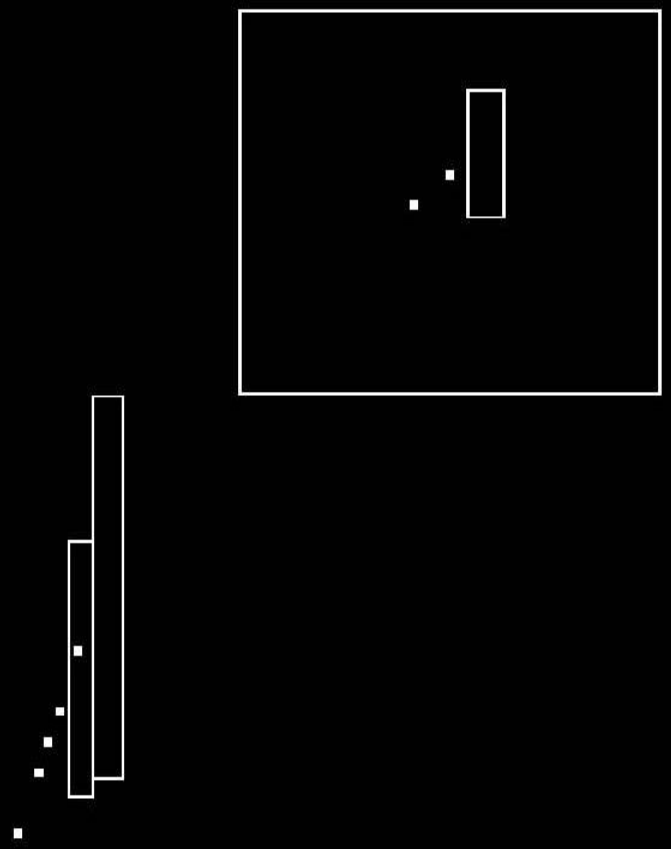



Original Image


명도 변화 감지


Simple Blob Filtering


Adjacent Blob Filtering



"후보 알고리즘"을 통한 Ball Tracking

Problems | Javadoc | Declaration | Console | LogCat | Error Log

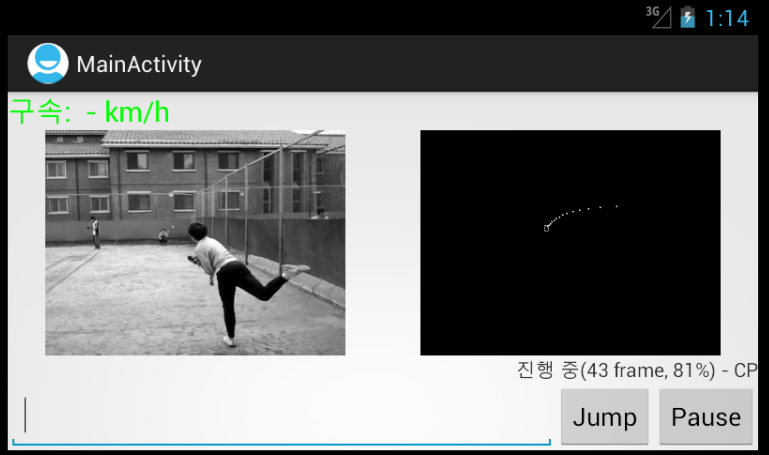
Main [Java Application] C:₩Program Files (x86)₩Java₩jre7₩bin₩javaw.exe (2012. 12. 3. 오전 10:13:38)

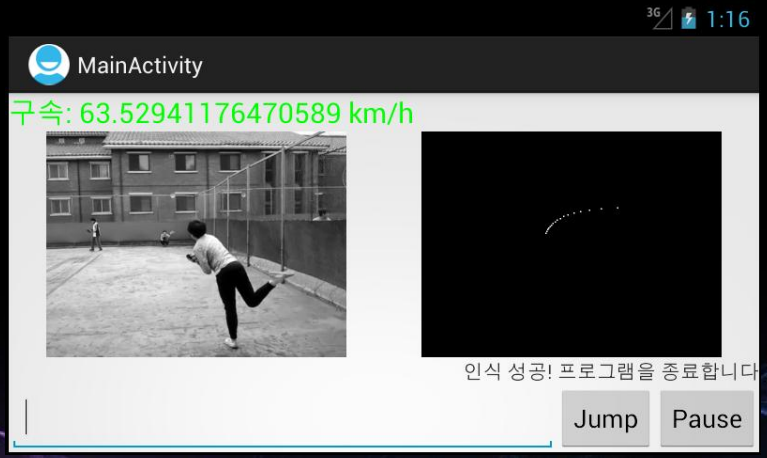
```
THERE ARE 1 CANDIDATES NOW
############## FRAME 343 ##############
[Blob_Labelling.java] Blob size = 33
BALL WAS CAUGHT /nf
The Speed of Pitch is 88km/h
ONLY one
ONLY one
NEW CANDIDATE WAS CREATED
ONLY one
ONLY one
NEW CANDIDATE WAS CREATED
THERE ARE 2 CANDIDATES NOW
############## FRAME 344 ##############
```

구속 계산


Catcher
포수 인식 및 S/B 판정 - 종료!

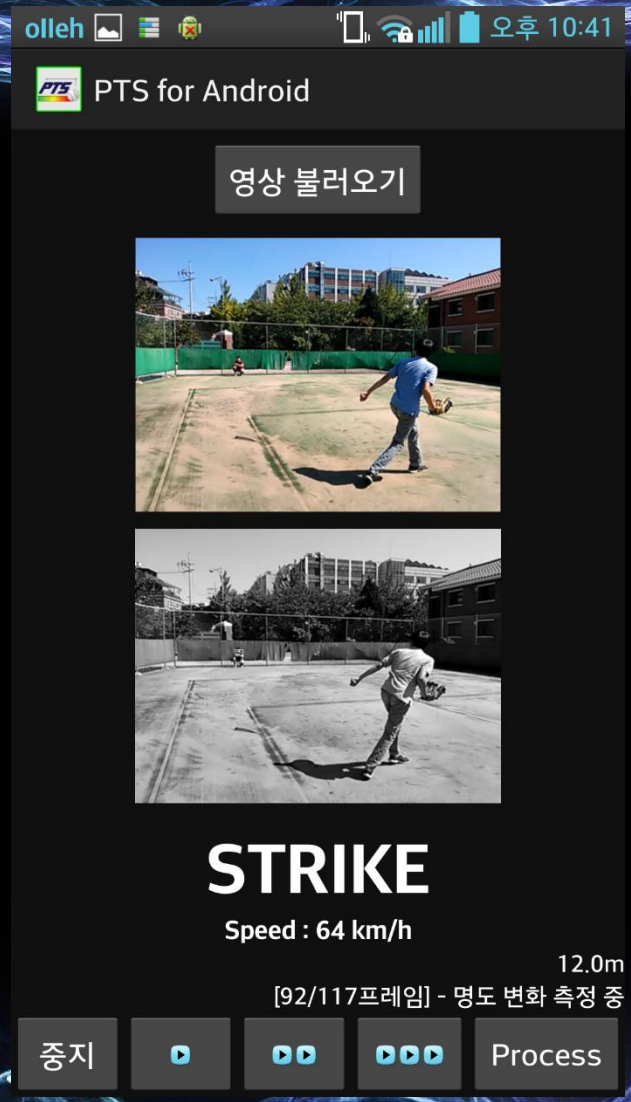
안드로이드 Prototype 앱





완성된 안드로이드 앱


olleh
오후 10:40
PTS for Android
영상 불러오기
12.0m
[5/117프레임]
중지
Process


olleh
오후 10:41
PTS for Android
영상 불러오기
STRIKE
Speed : 64 km/h
12.0m
[92/117프레임] - 명도 변화 측정 중
중지
Process

추적 정확도 평가

투수로부터 옆으로 떨어진 거리	0.7m	1.5m	2.2m
추적 성공률	78%	89%	100%

글러브의 색조	어두운 색	밝은 색
추적 성공률	89%	78%

높은 추적 성공률의 조건

배경

- 밝고 어두운 물체가 뒤섞인 복잡한 배경이 아닐 경우에는 평균적으로 충분히 양호

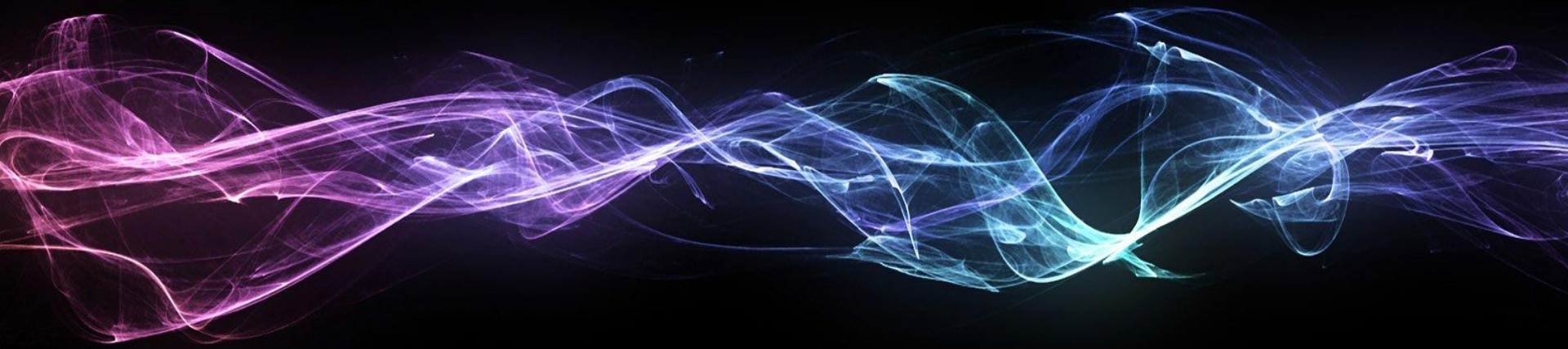
촬영 지점이 투수로부터 떨어진 수평거리

- 멀수록, 즉 프레임상에서 공의 수평방향 움직임이 클 수록

포수 글러브의 색조

- 공의 인식에 방해되지 않는 어두운 색조일 경우

Ⅴ. 결론



자체 평가

- 성공
  - **공 추적 및 구속 측정**
  - **포수 인식 및 스트라이크/볼 판정**
  - Java -> Android 포팅

- BUT
  - **실사용으로 느린 프로세싱 속도 (약 50s/회)**
  - 70%~80%의 다소 불안정한 정확도

알고리즘 개선방안

1 영상의 다운사이징

- 연산 횟수를 줄일 수 있는 가장 기본적인 방법 (현 640x480 -> 320x240)
- BUT **영상의 왜곡 발생 가능**

2 Histogram stretching

- 영상의 특정 부분에서 명도의 차를 증폭시켜 더 정밀한 thresholding 시도

3 사용자의 수동 조작 옵션

- 정확도가 시원치 않으면 사용자가 포수 위치 설정을 할 수 있도록 '매뉴얼모드' 포함
- 영상 왜곡이 발생하지 않음
- 그러나 사용자 편의성에 부정적 영향

4 서버 컴퓨팅

- 프로세서로 직접 처리하는 대신 인터넷을 통해 고성능 서버 컴퓨터에서 일괄적으로 계산을 담당
- 네트워크 의존성, 유지/관리 비용 발생

앞으로의 발전 방향

1. 포수 인식 - 기계 학습 알고리즘!

- 인식 과정 중 정확도에 가장 문제가 있는 부분이 포수 인식
- 기계학습 및 Pattern matching을 통한 더 정교한 방법 필요

2. Facebook 등 SNS와의 연동!

- 내 투구 결과를 SNS로 Share할 수 있는 기능을 추가하면
- 사용자들이 경쟁을 통해 새로운 Game 플랫폼 구축 가능

Open Source로서의 발전 가치

- 영상 분석에 대한 연구자들의 얄팍한 지식으로 인해 다소 무식한 계산 부분이 많음
  - OpenCV 권위자들의 지원이 필요
- 앞선 SNS 연동, 기계 학습 등 생소한 분야에 대한 개발 지원을 기대
- 무료 앱 배포로 더 많은 사람들의 야구 활동 참여를 기원!
- **(및 Github forking 좀)**


감사합니다.