Config 설정 정의 및 모드별 셋팅 가이드

개요

Recognizer 시스템은 `config.yaml` 파일을 통해 모든 모듈의 설정을 통합 관리합니다. 이 문서는 각 설정 항목의 의미와 모드별 최적 설정을 설명합니다.

전체 설정 구조

```yaml
# 기본 실행 모드
mode: inference.realtime

# 듀얼 서비스 설정
dual_service:
  enabled: false
  services: [falldown]

# 단계별 처리 설정 (Annotation)
annotation:
  input: /path/to/input
  output_dir: output
  pipeline_mode: true
  stage1: {...}
  stage2: {...}
  stage3: {...}

# 추론 설정 (Inference)
inference:
  analysis: {...}
  realtime: {...}
  visualize: {...}

# 모델 설정
models:
  pose_estimation: {...}
  fight_classification: {...}
  falldown_classification: {...}
  tracking: {...}
  scoring: {...}

# 성능 최적화
performance: {...}

# 이벤트 관리
events: {...}

# 파일 관리
files: {...}

# 로깅
logging: {...}

# 에러 처리
error_handling: {...}
```

1. 기본 모드 설정

1.1 실행 모드 선택

```yaml
# Annotation 모드 (데이터 준비)
mode: annotation.stage1      # 포즈 추정
mode: annotation.stage2      # 추적 및 스코어링
mode: annotation.stage3      # 데이터셋 생성
mode: annotation.visualize   # 결과 시각화

# Inference 모드 (실시간 처리)
mode: inference.analysis     # 배치 분석
mode: inference.realtime     # 실시간 처리
mode: inference.visualize    # PKL 결과 시각화
```

1.2 듀얼 서비스 설정

```yaml
dual_service:
  enabled: true              # 다중 서비스 동시 실행
  services:
    - fight                  # 폭력 감지
    - falldown               # 낙상 감지

# 단일 서비스 모드
dual_service:
  enabled: false
  services: [fight]          # 하나의 서비스만 사용
```

2. Annotation 모드 설정

2.1 기본 설정

```yaml
annotation:
  input: /path/to/videos              # 입력 비디오 디렉토리
  output_dir: output                  # 출력 디렉토리
  pipeline_mode: true                 # 파이프라인 모드 활성화
```

2.2 Stage1 설정 (포즈 추정)

```yaml
annotation:
  stage1:
    enabled: true
    # 멀티프로세스 설정
    multi_process:
      enabled: true                   # 멀티프로세스 활성화
      num_processes: 6                # 프로세스 수
      gpus: [0, 1]                    # GPU 목록 (라운드 로빈)
      chunk_strategy: video_split     # 분할 전략
```

프로세스 수 결정 기준:

- `CPU 코어 수 / 2` 정도가 적정
- GPU당 2-3개 프로세스 권장
- 메모리 사용량 고려 필요

GPU 할당 전략:

- `gpus: [0]` : 단일 GPU 사용
- `gpus: [0, 1]` : 두 GPU에 라운드 로빈 할당
- 프로세스들이 자동으로 GPU 순환 할당

2.3 Stage2 설정 (추적)

```yaml
annotation:
  stage2:
    enabled: true
    # Stage1 결과 자동 로드
```

입력 데이터:

- Stage1에서 생성된 PKL 파일들 자동 검색
- `*_stage1_poses.pkl` 패턴 매칭

2.4 Stage3 설정 (데이터셋)

```yaml
annotation:
  stage3:
    enabled: true
    split_ratios:
      train: 0.7                    # 훈련 데이터 비율
      val: 0.2                      # 검증 데이터 비율
      test: 0.1                     # 테스트 데이터 비율
```

분할 전략:

- 비디오 단위로 분할 (프레임 단위 아님)
- 클래스 균형 고려한 stratified split
- 시드 고정으로 재현 가능한 분할

2.5 시각화 설정

```yaml
annotation:
  visualize:
    stage: stage2                    # stage1 또는 stage2
    video_dir: /path/to/videos       # 원본 비디오 디렉토리
    results_dir: /path/to/results    # PKL 결과 디렉토리
    save_mode: false                 # 비디오 저장 여부
```

3. Inference 모드 설정

3.1 Analysis 모드 (배치 분석)

```yaml
inference:
  analysis:
    input: /path/to/test_videos      # 테스트 비디오 경로
    output_dir: output               # 결과 저장 경로

    # 성능평가 설정
    enable_evaluation: true          # 성능평가 활성화

    # 멀티프로세스 설정
    multi_process:
      enabled: true                  # 멀티프로세스 활성화
      num_processes: 4               # 프로세스 수
      gpus: [0, 1]                   # GPU 목록
      chunk_strategy: video_split    # 분할 전략
```

성능평가 기능:

- 혼동 행렬 (Confusion Matrix)
- 분류 보고서 (Precision, Recall, F1-score)
- 성능 차트 생성
- 최종 보고서 생성

3.2 Realtime 모드 (실시간 처리)

```yaml
inference:
  realtime:
    input: /path/to/video           # 비디오 파일 또는 디렉토리
    output_path: output             # 결과 저장 경로
    save_output: true               # 결과 저장 여부
    display_width: 640              # 디스플레이 너비
    display_height: 480             # 디스플레이 높이
    overlay_mode: skeleton_only     # 오버레이 모드
```

오버레이 모드:

- `full` : 키포인트 + 바운딩박스 + 점수 표시
- `skeleton_only` : 스켈레톤만 표시
- `raw` : 원본 영상만 표시

3.3 Visualize 모드 (PKL 시각화)

```yaml
inference:
  visualize:
    input: /path/to/videos          # 원본 비디오 경로
    results_dir: /path/to/pkls      # PKL 결과 경로
    save_dir: output                # 오버레이 비디오 저장 경로
    save_mode: true                 # 저장 여부
```

4. 모델 설정

4.1 포즈 추정 모델

ONNX 설정 (권장)

```yaml
models:
  pose_estimation:
    inference_mode: onnx            # 추론 모드
    onnx:
      model_path: /path/to/end2end_l.onnx
      device: cuda:0
      score_threshold: 0.3          # 사람 감지 임계값
      nms_threshold: 0.45           # NMS 임계값
      keypoint_threshold: 0.3       # 키포인트 신뢰도 임계값
      max_detections: 100           # 최대 감지 수
      model_input_size: [640, 640]  # 모델 입력 크기

      # ONNX Runtime 최적화
      execution_mode: ORT_SEQUENTIAL
      graph_optimization_level: ORT_ENABLE_ALL
      enable_mem_pattern: true
      enable_cpu_mem_arena: true
      gpu_mem_limit_gb: 8
```

PyTorch 설정

```yaml
models:
  pose_estimation:
    inference_mode: pth
    pth:
      config_file: /path/to/config.py
      checkpoint_path: /path/to/checkpoint.pth
      device: cuda:0
      score_threshold: 0.2
      nms_threshold: 0.65
      keypoint_threshold: 0.3
      input_size: [640, 640]
```

TensorRT 설정 (최고 성능)

```yaml
models:
  pose_estimation:
    inference_mode: tensorrt
    tensorrt:
      model_path: /path/to/model.trt
      device: cuda:0
      score_threshold: 0.3
      fp16_mode: false              # FP16 사용 여부
```

4.2 동작 분류 모델

Fight 분류기

```yaml
models:
  fight_classification:
    model_name: stgcn
    config_file: /path/to/stgcn_config.py
    checkpoint_path: /path/to/fight_model.pth
    class_names: [NonFight, Fight]
    num_classes: 2
    confidence_threshold: 0.4       # 분류 신뢰도 임계값
    device: cuda:0

    # STGCN 특화 설정
    input_format: stgcn
    coordinate_dimensions: 2        # 2D 좌표
    expected_keypoint_count: 17     # COCO17 형식
    max_persons: 4                  # 최대 인원
    window_size: 100               # 윈도우 크기 (프레임)
```

Falldown 분류기

```yaml
models:
  falldown_classification:
    model_name: stgcn
    config_file: /path/to/falldown_config.py
    checkpoint_path: /path/to/falldown_model.pth
    class_names: [Normal, Falldown]
    confidence_threshold: 0.4
    # ... 나머지 설정 동일
```

4.3 추적 설정

```yaml
models:
  tracking:
    tracker_name: bytetrack
    frame_rate: 30                  # 비디오 FPS

    # ByteTracker 파라미터
    track_thresh: 0.2               # 추적 시작 임계값
    match_thresh: 0.5               # 매칭 임계값
    track_high_thresh: 0.4          # 고신뢰도 추적 임계값
    track_low_thresh: 0.1           # 저신뢰도 추적 임계값
    new_track_thresh: 0.5           # 새 추적 생성 임계값
    track_buffer: 120               # 추적 버퍼 크기 (프레임)

    # 매칭 전략
    use_hybrid_matching: true       # 하이브리드 매칭 사용
    iou_weight: 0.7                 # IoU 가중치
    keypoint_weight: 0.3            # 키포인트 가중치
    min_box_area: 100               # 최소 박스 크기
    mot20: false                    # MOT20 데이터셋 모드
```

4.4 점수 계산 설정

Fight 스코어링

```yaml
models:
  scoring:
    fight:
      scorer_name: region_based     # motion_based와 동일
      min_track_length: 10          # 최소 추적 길이
      quality_threshold: 0.3        # 품질 임계값
      weights:
        movement: 0.4               # 움직임 가중치
        interaction: 0.4            # 상호작용 가중치
        position: 0.1               # 위치 가중치
        temporal: 0.1               # 시간적 가중치
```

Falldown 스코어링

```yaml
models:
  scoring:
    falldown:
      scorer_name: falldown_scorer
      min_track_length: 10
      quality_threshold: 0.3
      weights:
        height_change: 0.35         # 높이 변화
        posture_angle: 0.25         # 자세 각도
        movement_intensity: 0.20    # 움직임 강도
        persistence: 0.15           # 지속성
        position: 0.05              # 위치
```

5. 성능 최적화 설정

```yaml
performance:
  batch_size: 8                     # 배치 크기
  device: cuda:0                    # 기본 디바이스
  max_cache_size: 1000              # 캐시 크기
  window_size: 100                  # 윈도우 크기
  window_stride: 50                 # 윈도우 스트라이드

  # 메모리 관리
  enable_garbage_collection: true   # 가비지 컬렉션 활성화
  gc_interval: 100                  # GC 실행 간격 (프레임)
```

배치 크기 조정 기준:

- GPU 메모리에 따라 조정 (RTX 3090: 16, RTX 4090: 32)
- 메모리 부족 시 점진적 감소

6. 이벤트 관리 설정

6.1 Fight 이벤트

```yaml
events:
  fight:
    alert_threshold: 0.8            # 알림 임계값
    min_consecutive_detections: 3   # 최소 연속 감지 횟수
    normal_threshold: 0.5           # 정상 상태 임계값
    min_consecutive_normal: 2       # 최소 연속 정상 횟수
    min_event_duration: 2.0         # 최소 이벤트 지속시간 (초)
    max_event_duration: 10.0        # 최대 이벤트 지속시간 (초)
    cooldown_duration: 5.0          # 쿨다운 시간 (초)
```

6.2 Falldown 이벤트

```yaml
events:
  falldown:
    alert_threshold: 0.6            # 낮은 임계값 (빠른 감지)
    min_consecutive_detections: 2   # 빠른 반응
    normal_threshold: 0.4
    min_consecutive_normal: 3       # 확실한 해제
    min_event_duration: 1.0         # 빠른 감지
    max_event_duration: 15.0        # 긴 지속시간
    cooldown_duration: 3.0          # 짧은 쿨다운
```

6.3 공통 이벤트 설정

```yaml
events:
  # 알림 설정
  enable_ongoing_alerts: true       # 진행중 알림 활성화
  ongoing_alert_interval: 30.0      # 진행중 알림 간격 (초)

  # 로그 설정
  save_event_log: true              # 이벤트 로그 저장
  event_log_format: json           # 로그 형식 (json/csv)
  event_log_path: output/event_logs # 로그 저장 경로
```

7. 파일 관리 설정

```yaml
files:
  # 출력 구조
  output_structure:
    pkl_dir: pkl                    # PKL 파일 디렉토리
    json_dir: json                  # JSON 파일 디렉토리
    overlay_dir: overlay            # 오버레이 비디오 디렉토리

  # 파일명 규칙
  naming:
    include_timestamp: false        # 타임스탬프 포함 여부
    poses_suffix: _frame_poses      # 포즈 파일 접미사
    results_suffix: _results        # 결과 파일 접미사
    rtmo_suffix: _rtmo_poses        # RTMO 파일 접미사

  # 지원 비디오 형식
  video_extensions:
    - .mp4
    - .avi
    - .mov
    - .mkv
    - .flv
    - .wmv
```

8. 로깅 및 에러 처리

8.1 로깅 설정

```yaml
logging:
  level: INFO                       # DEBUG, INFO, WARNING, ERROR
  format: '%(asctime)s - %(levelname)s - %(message)s'
```

로그 레벨 가이드:

- `DEBUG` : 개발/디버깅 시 사용
- `INFO` : 일반적인 운영 상황
- `WARNING` : 잠재적 문제 상황
- `ERROR` : 오류 발생 상황만

8.2 에러 처리

```yaml
error_handling:
  continue_on_error: true           # 에러 발생시 계속 진행
  error_recovery_strategy: skip     # skip, retry, abort
  max_consecutive_errors: 10        # 최대 연속 에러 수
```

에러 복구 전략:

- `skip` : 실패한 파일/프레임 건너뛰기
- `retry` : 재시도 후 실패시 건너뛰기
- `abort` : 에러 발생시 전체 중단

9. 모드별 권장 설정

9.1 개발/디버깅 설정

```yaml
# 개발용 설정
mode: inference.realtime
dual_service:
  enabled: false
  services: [fight]

logging:
  level: DEBUG

performance:
  batch_size: 4                    # 작은 배치 크기

inference:
  realtime:
    overlay_mode: full             # 모든 정보 표시
    save_output: true
```

9.2 프로덕션 설정

```yaml
# 프로덕션용 설정
mode: inference.realtime
dual_service:
  enabled: true
  services: [fight, falldown]

logging:
  level: WARNING                   # 경고 이상만 로깅

performance:
  batch_size: 16                   # 큰 배치 크기
  enable_garbage_collection: true

models:
  pose_estimation:
    inference_mode: onnx           # ONNX 사용 (빠름)

error_handling:
  continue_on_error: true          # 안정성 우선
```

9.3 대용량 데이터 처리 설정

```yaml
# 대용량 배치 처리
mode: annotation.stage1

annotation:
  stage1:
    multi_process:
      enabled: true
      num_processes: 12              # 많은 프로세스
      gpus: [0, 1, 2, 3]            # 다중 GPU

performance:
  batch_size: 32                     # 큰 배치
  enable_garbage_collection: true
  gc_interval: 50                    # 자주 GC

logging:
  level: WARNING                     # 최소 로깅
```

10. 설정 검증 및 최적화

10.1 설정 검증

시스템은 자동으로 설정 유효성을 검증합니다:

- 필수 필드 존재 여부
- 파일 경로 유효성
- GPU 메모리 충분성
- 모델 호환성

10.2 성능 튜닝 가이드

GPU 메모리 최적화

```yaml
# 메모리 부족시
performance:
  batch_size: 4                      # 배치 크기 감소

models:
  pose_estimation:
    onnx:
      gpu_mem_limit_gb: 4            # GPU 메모리 제한
```

처리 속도 최적화

```yaml
# 속도 우선시
models:
  pose_estimation:
    inference_mode: onnx            # 또는 tensorrt
    onnx:
      score_threshold: 0.5          # 높은 임계값 (검출 수 감소)

performance:
  window_stride: 100               # 큰 스트라이드 (겹침 감소)
```