Stage별 PKL 데이터 구조 설명서

개요

Recognizer 프로젝트는 3단계 파이프라인을 통해 비디오를 처리하며, 각 단계마다 특화된 PKL 데이터 구조를 생성합니다.

전체 처리 흐름:

```
원본 비디오 → Stage1 (포즈 추정) → Stage2 (추적) → Stage3 (데이터셋)
```

Stage1: 포즈 추정 PKL 구조

파일 명명 규칙

```
{video_name}_stage1_poses.pkl
```

저장 위치

```
output/{dataset_name}/stage1_poses/{estimator_config}/
├── video1_stage1_poses.pkl
├── video2_stage1_poses.pkl
└── ...
```

데이터 구조

최상위 구조체 (VisualizationData)

```python
@dataclass
class VisualizationData:
    video_name: str                    # 비디오 파일명 (확장자 제외)
    frame_data: List[FramePoses]       # 프레임별 포즈 데이터
    stage_info: Dict[str, Any]         # 단계별 메타데이터
    poses_only: List[FramePoses]       # frame_data와 동일 (호환성)
    poses_with_tracking: None          # Stage2에서 활용
    tracking_info: None                # Stage2에서 활용
    poses_with_scores: None            # 확장 가능
    scoring_info: None                 # 확장 가능
    classification_results: None       # 확장 가능
```

FramePoses 구조

```python
@dataclass
class FramePoses:
    frame_idx: int                     # 프레임 인덱스 (0부터 시작)
    persons: List[Person]              # 프레임 내 감지된 사람들
    timestamp: float                   # 프레임 타임스탬프 (초)
    image_shape: Tuple[int, int]       # 이미지 크기 (height, width)
    metadata: Dict[str, Any] = None    # 추가 메타데이터
```

Person 구조 (Stage1)

```
@dataclass
class Person:
    person_id: int                      # 프레임 내 사람 ID (고유)
    bbox: Tuple[float, float, float, float]  # 바운딩박스 (x1, y1, x2, y2)
    keypoints: np.ndarray               # 키포인트 좌표 [17, 2]
    score: float                        # 사람 감지 신뢰도 점수
    track_id: None                      # Stage1에서는 None (미할당)
    timestamp: float                    # 프레임 타임스탬프
    metadata: Dict[str, Any] = None     # 추가 메타데이터
```

stage_info 구조

```
{
    'stage': 'pose_estimation',
    'total_frames': int,                # 총 프레임 수
    'config': {                         # 포즈 추정 설정
        'model_name': 'rtmo_onnx',
        'score_threshold': 0.3,
        'nms_threshold': 0.45,
        'keypoint_threshold': 0.3,
        'model_path': '/workspace/mmpose/checkpoints/end2end_l.onnx'
    },
    'original_path': str,               # 원본 비디오 파일 경로
    'original_label': int               # 폴더 기반 라벨 (0: NonFight, 1: Fight)
}
```

키포인트 형식

- **포맷**: COCO17 (17개 관절점)
- **좌표**: 픽셀 좌표 (x, y)
- **순서**:

```
0: nose,         1: left_eye,     2: right_eye,    3: left_ear,     4: right_ear,
5: left_shoulder, 6: right_shoulder, 7: left_elbow,  8: right_elbow,
9: left_wrist,   10: right_wrist, 11: left_hip,    12: right_hip,
13: left_knee,   14: right_knee,  15: left_ankle,  16: right_ankle
```

데이터 접근 예시

```
import pickle

# Stage1 데이터 로드
with open('video1_stage1_poses.pkl', 'rb') as f:
    data = pickle.load(f)

# 기본 정보
video_name = data.video_name
total_frames = len(data.frame_data)

# 첫 번째 프레임의 첫 번째 사람
first_frame = data.frame_data[0]
if first_frame.persons:
    person = first_frame.persons[0]
    keypoints = person.keypoints  # [17, 2] 배열
    bbox = person.bbox           # (x1, y1, x2, y2)
    confidence = person.score    # 신뢰도
```

Stage2: 추적 및 스코어링 PKL 구조

파일 명명 규칙

```
{video_name}_stage2_tracking.pkl
```

저장 위치

```
output/{dataset_name}/stage2_tracking/{tracker_config}/
├── video1_stage2_tracking.pkl
├── video2_stage2_tracking.pkl
└── ...
```

데이터 구조

최상위 구조체 (VisualizationData)

```
@dataclass
class VisualizationData:
    video_name: str                    # 비디오 파일명
    frame_data: List[FramePoses]       # 원본 프레임 데이터 (참조용)
    stage_info: Dict[str, Any]         # 단계 메타데이터
    poses_only: None                   # Stage1에서만 사용
    poses_with_tracking: List[FramePoses]  # 추적이 적용된 데이터
    tracking_info: Dict[str, Any]      # 추적 관련 정보
    poses_with_scores: None            # 확장 가능
    scoring_info: None                 # 확장 가능
    classification_results: None       # 확장 가능
```

Person 구조 (Stage2 - 추적 정보 추가)

```
@dataclass
class Person:
    person_id: int                      # 프레임 내 사람 ID
    bbox: Tuple[float, float, float, float]  # 바운딩박스
    keypoints: np.ndarray               # 키포인트 좌표 [17, 2]
    score: float                        # 감지 신뢰도
    track_id: int                       # 추적 ID (비디오 전체에서 고유)
    timestamp: float                    # 타임스탬프
    metadata: Dict[str, Any]            # 스코어링 정보 포함
```

stage_info 구조 (Stage2)

```
{
    'stage': 'tracking_scoring',
    'total_frames': int,
    'tracking_config': {                # ByteTracker 설정
        'track_thresh': 0.2,
        'match_thresh': 0.5,
        'track_high_thresh': 0.4,
        'track_low_thresh': 0.1,
        'new_track_thresh': 0.5,
        'track_buffer': 120,
        'frame_rate': 30,
        'mot20': false
    },
    'scoring_config': {                 # 스코어링 설정
        'scorer_name': 'region_based',
        'quality_threshold': 0.3,
        'min_track_length': 10,
        'weights': {
            'movement': 0.4,
            'interaction': 0.4,
            'position': 0.1,
            'temporal': 0.1
        }
    },
    'original_path': str,               # 원본 경로 보존
    'original_label': int               # 원본 라벨 보존
}
```

tracking_info 구조

```
{
    'total_tracks': int,                # 생성된 총 추적 수
    'active_tracks': int,               # 최종 활성 추적 수
    'config': Dict,                     # 추적기 설정 정보
    'track_statistics': {               # 추적 통계
        'avg_track_length': float,
        'max_track_length': int,
        'min_track_length': int
    }
}
```

데이터 접근 예시

```
# Stage2 데이터 로드
with open('video1_stage2_tracking.pkl', 'rb') as f:
    data = pickle.load(f)

# 추적된 데이터 접근
tracked_frames = data.poses_with_tracking
total_tracks = data.tracking_info['total_tracks']

# 특정 프레임의 추적된 사람들
frame = tracked_frames[100]
for person in frame.persons:
    track_id = person.track_id  # 비디오 전체에서 고유한 ID
    if person.metadata:
        # 스코어링 정보 (있는 경우)
        movement_score = person.metadata.get('movement_score', 0.0)
```

Stage3: 최종 데이터셋 PKL 구조

파일 구조

```
output/{dataset_name}/stage3_dataset/{full_config_name}/
├── train.pkl                        # 훈련 데이터
├── val.pkl                          # 검증 데이터
├── test.pkl                         # 테스트 데이터
├── metadata.json                    # 데이터셋 메타데이터
└── stage3_path_info.json           # 경로 정보
```

PKL 파일 구조

각 PKL 파일은 **샘플 딕셔너리의 리스트**입니다:

```
List[Dict[str, Any]]  # 샘플들의 리스트
```

샘플 구조 (STGCN 호환)

```
{
    'frame_dir': str,                 # 프레임 식별자 (예: "V_960")
    'total_frames': int,              # 총 프레임 수
    'img_shape': (int, int),          # 이미지 크기 (H, W)
    'original_shape': (int, int),     # 원본 크기 (H, W)
    'label': int,                     # 클래스 라벨 (0: NonFight, 1: Fight)
    'keypoint': np.ndarray,           # [M, T, V, C] 키포인트 텐서
    'keypoint_score': np.ndarray      # [M, T, V] 신뢰도 텐서
}
```

텐서 차원 상세

keypoint 텐서: `[M, T, V, C]`

- **M**: 최대 사람 수 (4명으로 제한)

- **T**: 시간 프레임 수 (비디오 길이)
- **V**: 키포인트 수 (17개, COCO17)
- **C**: 좌표 차원 (2차원: x, y)

keypoint_score 텐서: `[M, T, V]`

- **M**: 최대 사람 수 (4명)
- **T**: 시간 프레임 수
- **V**: 키포인트 수 (17개)

데이터 범위 및 특성

- **keypoint 좌표**: 픽셀 좌표 (예: -18.58 ~ 640.0)
- **keypoint_score**: 신뢰도 점수 (0.0 ~ 1.0)
- **label**: 0 (정상) 또는 1 (이상)
- **패딩**: 4명 미만인 경우 0으로 패딩
- **시간 정규화**: 모든 비디오가 동일한 시간 길이로 정규화되지 않음

metadata.json 구조

```
{
    "dataset_info": {
        "name": "dataset_name",
        "total_samples": 5694,
        "total_videos": 5694,
        "failed_files": 253,
        "success_rate": 95.7
    },
    "split_info": {
        "train_count": 3985,
        "val_count": 1138,
        "test_count": 571,
        "split_ratios": [0.7, 0.2, 0.1]
    },
    "class_distribution": {
        "class_0": 2583,  // NonFight
        "class_1": 3111   // Fight
    },
    "processing_config": {
        "stage1_config": {...},
        "stage2_config": {...},
        "stage3_config": {...}
    }
}
```

데이터 접근 예시

```
# Stage3 데이터 로드
with open('train.pkl', 'rb') as f:
    train_data = pickle.load(f)

# 첫 번째 샘플
sample = train_data[0]
keypoints = sample['keypoint']      # [4, T, 17, 2]
scores = sample['keypoint_score']   # [4, T, 17]
label = sample['label']             # 0 or 1
video_name = sample['frame_dir']    # "V_960"

# 첫 번째 사람의 첫 번째 프레임 키포인트
person1_frame1 = keypoints[0, 0, :, :]  # [17, 2]

# 해당 키포인트들의 신뢰도
person1_frame1_scores = scores[0, 0, :]  # [17]
```

데이터 변환 과정

Stage1 → Stage2 변환

- Person 객체에 `track_id` 필드 추가
- `poses_with_tracking` 필드에 추적 결과 저장
- `tracking_info` 에 추적 통계 정보 저장

Stage2 → Stage3 변환

- 프레임별 Person 리스트를 텐서로 변환
- 최대 4명까지 제한, 부족하면 0 패딩
- STGCN 모델에 직접 입력 가능한 형식으로 변환
- 비디오별로 하나의 샘플 딕셔너리 생성

데이터셋 통계 (예시: RWF-2000)

최종 데이터셋 구성

- **총 샘플**: 5,694개 비디오
- **훈련 세트**: 3,985개 (70%)
- **검증 세트**: 1,138개 (20%)
- **테스트 세트**: 571개 (10%)

클래스 분포

- **NonFight(0)**: 2,583개 (45.4%)
- **Fight(1)**: 3,111개 (54.6%)

처리 성공률

- **처리 성공**: 5,694개
- **처리 실패**: 253개 (4.3%)

- **전체 성공률**: 95.7%

주요 설계 원칙

1. **라벨 일관성**: 원본 비디오의 클래스 라벨이 모든 단계에서 보존
2. **추적 연속성**: Stage2에서 비디오 전체에 걸쳐 일관된 ID 할당
3. **MMAction2 호환**: Stage3 출력이 MMAction2 STGCN과 직접 호환
4. **메타데이터 보존**: 각 단계의 설정과 통계 정보 완전 보존
5. **확장성**: 새로운 필드 추가 및 다양한 모델 지원 가능한 구조
6. **디버깅 지원**: 실패 정보와 중간 결과 추적 가능