API 참조 가이드

개요

Recognizer 시스템의 모든 API 엔드포인트, 클래스, 메서드에 대한 상세한 참조 문서입니다. 각 모드별 API 사용법과 예제 코드를 포함하여 개발자가 시스템을 효율적으로 활용할 수 있도록 안내합니다.

목차

1. Core API

1.1 Main Entry Point

`recognizer.main`

시스템의 메인 진입점으로 모든 모드의 실행을 관리합니다.

```
def main(args: Optional[List[str]] = None) -> int:
    """
    Recognizer 시스템 메인 실행 함수

    Args:
        args: 명령행 인수 리스트 (None이면 sys.argv 사용)

    Returns:
        int: 종료 코드 (0: 성공, 1: 실패)

    Example:
        >>> from recognizer.main import main
        >>> exit_code = main(['--mode', 'inference.realtime'])
        >>> print(f"Exit code: {exit_code}")
    """
```

Command Line Interface

```
# 기본 사용법
python main.py --mode <mode_name> [options]

# 실시간 추론 실행
python main.py --mode inference.realtime --log-level INFO

# 배치 분석 실행
python main.py --mode inference.analysis --input-dir /path/to/videos

# Annotation Stage1 실행
python main.py --mode annotation.stage1 --input-dir /path/to/videos

# 설정 파일 지정
python main.py --config /path/to/custom_config.yaml

# 도움말 출력
python main.py --help
```

1.2 Core Classes

`BaseMode`

모든 실행 모드의 기본 클래스입니다.

```
class BaseMode(ABC):
    """모든 실행 모드의 추상 기본 클래스"""

    def __init__(self, config: Config):
        """
        모드 초기화

        Args:
            config: 설정 객체
        """
        self.config = config
        self.logger = self._setup_logger()

    @abstractmethod
    def execute(self) -> ExecutionResult:
        """
        모드 실행 추상 메서드

        Returns:
            ExecutionResult: 실행 결과 객체

        Raises:
            NotImplementedError: 서브클래스에서 구현 필요
        """
        pass

    def _setup_logger(self) -> logging.Logger:
        """로거 설정"""
        pass

    def _validate_config(self) -> bool:
        """설정 검증"""
        pass
```

`ExecutionResult`

실행 결과를 담는 데이터 클래스입니다.

```python
@dataclass
class ExecutionResult:
    """실행 결과 데이터 클래스"""

    success: bool
    message: str
    execution_time: float
    processed_items: int
    output_paths: List[str]
    performance_metrics: Dict[str, Any]
    error_details: Optional[str] = None

    def to_dict(self) -> Dict[str, Any]:
        """딕셔너리로 변환"""
        return asdict(self)

    def save_to_file(self, filepath: str) -> None:
        """결과를 파일로 저장"""
        with open(filepath, 'w') as f:
            json.dump(self.to_dict(), f, indent=2)
```

2. Annotation Mode APIs

2.1 Stage1: Pose Estimation API

`Stage1Mode`

```python
class Stage1Mode(BaseMode):
    """Stage1 포즈 추정 모드"""

    def __init__(self, config: Stage1Config):
        super().__init__(config)
        self.pose_estimator = self._create_pose_estimator()
        self.video_processor = VideoProcessor(config.video_config)

    def execute(self) -> ExecutionResult:
        """
        Stage1 실행: 비디오에서 포즈 추정

        Returns:
            ExecutionResult: 처리 결과

        Example:
            >>> config = Stage1Config.from_yaml('config.yaml')
            >>> stage1 = Stage1Mode(config)
            >>> result = stage1.execute()
            >>> print(f"Processed {result.processed_items} videos")
        """

    def process_video(self, video_path: str) -> PoseEstimationResult:
        """
        단일 비디오 처리

        Args:
            video_path: 비디오 파일 경로

        Returns:
            PoseEstimationResult: 포즈 추정 결과

        Raises:
            VideoLoadError: 비디오 로드 실패
            PoseEstimationError: 포즈 추정 실패
        """

    def batch_process_videos(self, video_paths: List[str]) -> List[PoseEstimationResult]:
        """
        다중 비디오 배치 처리

        Args:
            video_paths: 비디오 파일 경로 리스트

        Returns:
            List[PoseEstimationResult]: 처리 결과 리스트
        """
```

RTMOONNXEstimator

RTMO 포즈 추정기 클래스입니다.

```
class RTMOONNXEstimator:
    """RTMO ONNX 포즈 추정기"""

    def __init__(self, model_config: RTMOConfig):
        """
        RTMO 추정기 초기화

        Args:
            model_config: RTMO 모델 설정

        Example:
            >>> config = RTMOConfig(
            ...     model_path='models/rtmo-s_8xb32-600e_body7-640x640.onnx',
            ...     input_size=(640, 640),
            ...     confidence_threshold=0.3
            ... )
            >>> estimator = RTMOONNXEstimator(config)
        """

    def estimate(self, frame: np.ndarray) -> List[Person]:
        """
        프레임에서 포즈 추정

        Args:
            frame: 입력 프레임 (H, W, 3)

        Returns:
            List[Person]: 감지된 인물 리스트

        Example:
            >>> frame = cv2.imread('image.jpg')
            >>> persons = estimator.estimate(frame)
            >>> for person in persons:
            ...     print(f"Person confidence: {person.confidence}")
            ...     print(f"Keypoints shape: {person.keypoints.shape}")
        """

    def estimate_batch(self, frames: List[np.ndarray]) -> List[List[Person]]:
        """
        배치 프레임 포즈 추정

        Args:
            frames: 프레임 리스트

        Returns:
            List[List[Person]]: 프레임별 인물 리스트
        """

    def preprocess_frame(self, frame: np.ndarray) -> np.ndarray:
        """프레임 전처리"""

    def postprocess_results(self, outputs: np.ndarray) -> List[Person]:
        """결과 후처리"""
```

Data Structures

```python
@dataclass
class Person:
    """인물 데이터 클래스"""

    keypoints: np.ndarray      # Shape: (17, 3) - [x, y, confidence]
    bbox: Tuple[int, int, int, int]  # [x1, y1, x2, y2]
    confidence: float          # 전체 신뢰도
    person_id: Optional[int] = None    # 추적 ID (Stage2에서 할당)

    def get_keypoint(self, keypoint_name: str) -> np.ndarray:
        """
        키포인트 이름으로 좌표 가져오기

        Args:
            keypoint_name: 키포인트 이름 ('nose', 'left_eye', etc.)

        Returns:
            np.ndarray: [x, y, confidence]

        Example:
            >>> person = Person(keypoints=keypoints_array, bbox=bbox, confidence=0.9)
            >>> nose = person.get_keypoint('nose')
            >>> print(f"Nose position: {nose[:2]}")
        """

    def is_valid_pose(self, min_keypoints: int = 5) -> bool:
        """유효한 포즈인지 확인"""

    def get_center_point(self) -> Tuple[float, float]:
        """신체 중심점 계산"""

@dataclass
class FramePoses:
    """프레임 단위 포즈 데이터"""

    frame_number: int
    timestamp: float
    persons: List[Person]
    frame_shape: Tuple[int, int, int]  # (H, W, C)

    def get_person_count(self) -> int:
        """인물 수 반환"""
        return len(self.persons)

    def filter_by_confidence(self, min_confidence: float) -> 'FramePoses':
        """신뢰도 기준 필터링"""

    def to_dict(self) -> Dict[str, Any]:
        """딕셔너리로 변환"""
```

2.2 Stage2: Tracking API

`Stage2Mode`

```python
class Stage2Mode(BaseMode):
    """Stage2 추적 모드"""

    def __init__(self, config: Stage2Config):
        super().__init__(config)
        self.tracker = ByteTrackerWrapper(config.tracking_config)
        self.scorer = self._create_scorer()

    def execute(self) -> ExecutionResult:
        """Stage2 실행: 포즈 데이터에 추적 정보 추가"""

    def process_stage1_file(self, stage1_path: str) -> TrackingResult:
        """
        Stage1 PKL 파일 처리

        Args:
            stage1_path: Stage1 PKL 파일 경로

        Returns:
            TrackingResult: 추적 결과

        Example:
            >>> stage2 = Stage2Mode(config)
            >>> result = stage2.process_stage1_file('video_stage1_poses.pkl')
            >>> print(f"Tracked {len(result.tracked_sequences)} persons")
        """
```

ByteTrackerWrapper

```python
class ByteTrackerWrapper:
    """ByteTracker 래퍼 클래스"""

    def __init__(self, config: TrackingConfig):
        """
        ByteTracker 초기화

        Args:
            config: 추적 설정
        """

    def track(self, persons: List[Person]) -> List[TrackedPerson]:
        """
        인물 추적 수행

        Args:
            persons: 입력 인물 리스트

        Returns:
            List[TrackedPerson]: 추적된 인물 리스트

        Example:
            >>> tracker = ByteTrackerWrapper(config)
            >>> tracked_persons = tracker.track(persons)
            >>> for tp in tracked_persons:
            ...     print(f"Track ID: {tp.track_id}, State: {tp.track_state}")
        """

    def update_tracks(self, detections: List[Detection]) -> List[Track]:
        """추적 정보 업데이트"""

    def get_active_tracks(self) -> List[Track]:
        """활성 추적 정보 가져오기"""

    def reset(self) -> None:
        """추적기 리셋"""
```

MotionScorer

```python
class MotionScorer:
    """움직임 기반 점수 계산기"""

    def __init__(self, config: ScorerConfig):
        self.config = config
        self.velocity_weight = config.velocity_weight
        self.acceleration_weight = config.acceleration_weight

    def calculate_score(self, pose_sequence: List[TrackedPerson]) -> float:
        """
        포즈 시퀀스의 움직임 점수 계산

        Args:
            pose_sequence: 추적된 포즈 시퀀스

        Returns:
            float: 움직임 점수 (0.0 ~ 1.0)

        Example:
            >>> scorer = MotionScorer(config)
            >>> score = scorer.calculate_score(pose_sequence)
            >>> print(f"Motion score: {score:.3f}")
        """

    def calculate_velocity_features(self, keypoints_sequence: np.ndarray) -> np.ndarray:
        """속도 특징 계산"""

    def calculate_acceleration_features(self, keypoints_sequence: np.ndarray) -> np.ndarray:
        """가속도 특징 계산"""
```

2.3 Stage3: Dataset Creation API

`Stage3Mode`

```python
class Stage3Mode(BaseMode):
    """Stage3 데이터셋 생성 모드"""

    def __init__(self, config: Stage3Config):
        super().__init__(config)
        self.tensor_converter = TensorConverter(config.tensor_config)
        self.dataset_splitter = DatasetSplitter(config.split_config)

    def execute(self) -> ExecutionResult:
        """Stage3 실행: STGCN 호환 데이터셋 생성"""

    def create_dataset_from_stage2_files(self, stage2_paths: List[str]) -> DatasetCreationResult:
        """
        Stage2 파일들로부터 데이터셋 생성

        Args:
            stage2_paths: Stage2 PKL 파일 경로 리스트

        Returns:
            DatasetCreationResult: 데이터셋 생성 결과

        Example:
            >>> stage3 = Stage3Mode(config)
            >>> stage2_files = glob.glob('*_stage2_tracking.pkl')
            >>> result = stage3.create_dataset_from_stage2_files(stage2_files)
            >>> print(f"Created dataset with {result.total_samples} samples")
        """
```

TensorConverter

```python
class TensorConverter:
    """텐서 변환기"""

    def __init__(self, config: TensorConfig):
        """
        텐서 변환기 초기화

        Args:
            config: 텐서 변환 설정
        """
        self.max_persons = config.max_persons
        self.sequence_length = config.sequence_length
        self.num_keypoints = config.num_keypoints
        self.coord_dims = config.coord_dims

    def convert_to_stgcn_format(self, tracking_data: List[TrackingResult]) -> STGCNDataset:
        """
        추적 데이터를 STGCN 형식으로 변환

        Args:
            tracking_data: 추적 결과 데이터 리스트

        Returns:
            STGCNDataset: STGCN 호환 데이터셋

        Example:
            >>> converter = TensorConverter(config)
            >>> dataset = converter.convert_to_stgcn_format(tracking_data)
            >>> print(f"Dataset shape: {dataset.data.shape}")  # [N, M, T, V, C]
        """

    def normalize_coordinates(self, keypoints: np.ndarray, frame_shape: Tuple[int, int]) -> np.ndarray:
        """좌표 정규화"""

    def apply_temporal_padding(self, sequence: np.ndarray, target_length: int) -> np.ndarray:
        """시간적 패딩 적용"""

    def create_4d_tensor(self, pose_sequences: List[np.ndarray]) -> np.ndarray:
        """4D 텐서 생성 [M, T, V, C]"""
```

3. Inference Mode APIs

3.1 Analysis Mode API

`AnalysisMode`

```python
class AnalysisMode(BaseMode):
    """Analysis 배치 분석 모드"""

    def __init__(self, config: AnalysisConfig):
        super().__init__(config)
        self.pipeline = DualServicePipeline(config.pipeline_config)
        self.performance_analyzer = PerformanceAnalyzer()
        self.report_generator = ReportGenerator()

    def execute(self) -> ExecutionResult:
        """Analysis 모드 실행"""

    def analyze_video_batch(self, video_paths: List[str]) -> BatchAnalysisResult:
        """
        비디오 배치 분석

        Args:
            video_paths: 분석할 비디오 파일 경로 리스트

        Returns:
            BatchAnalysisResult: 배치 분석 결과

        Example:
            >>> analysis = AnalysisMode(config)
            >>> video_files = glob.glob('/path/to/videos/*.mp4')
            >>> result = analysis.analyze_video_batch(video_files)
            >>> print(f"Analyzed {result.total_videos} videos")
            >>> print(f"Average accuracy: {result.metrics.accuracy:.3f}")
        """

    def generate_performance_report(self, results: List[AnalysisResult]) -> PerformanceReport:
        """성능 보고서 생성"""

    def export_results(self, results: BatchAnalysisResult, output_dir: str) -> None:
        """결과 내보내기 (JSON, CSV, Charts)"""
```

PerformanceAnalyzer

```python
class PerformanceAnalyzer:
    """성능 분석기"""

    def __init__(self):
        self.metrics_calculator = MetricsCalculator()
        self.visualizer = ResultVisualizer()

    def analyze_classification_performance(self, predictions: List[ClassificationResult],
                                           ground_truth: List[str]) -> ClassificationMetrics:
        """
        분류 성능 분석

        Args:
            predictions: 예측 결과 리스트
            ground_truth: 정답 라벨 리스트

        Returns:
            ClassificationMetrics: 분류 성능 메트릭

        Example:
            >>> analyzer = PerformanceAnalyzer()
            >>> metrics = analyzer.analyze_classification_performance(predictions, labels)
            >>> print(f"Accuracy: {metrics.accuracy:.3f}")
            >>> print(f"F1 Score: {metrics.f1_score:.3f}")
            >>> print(f"Precision: {metrics.precision:.3f}")
            >>> print(f"Recall: {metrics.recall:.3f}")
        """

    def generate_confusion_matrix(self, predictions: List[str],
                                  ground_truth: List[str]) -> np.ndarray:
        """혼동 행렬 생성"""

    def calculate_temporal_metrics(self, results: List[AnalysisResult]) -> TemporalMetrics:
        """시간적 성능 메트릭 계산"""

    def create_performance_visualizations(self, metrics: ClassificationMetrics,
                                          output_dir: str) -> None:
        """성능 시각화 생성"""
```

3.2 Realtime Mode API

`RealtimeMode`

```python
class RealtimeMode(BaseMode):
    """실시간 추론 모드"""

    def __init__(self, config: RealtimeConfig):
        super().__init__(config)
        self.pipeline = DualServicePipeline(config.pipeline_config)
        self.visualizer = RealtimeVisualizer(config.visualization_config)
        self.event_manager = EventManager(config.event_config)

    def execute(self) -> ExecutionResult:
        """실시간 모드 실행"""

    def start_realtime_processing(self, input_source: str) -> None:
        """
        실시간 처리 시작

        Args:
            input_source: 입력 소스 (카메라 ID, 비디오 파일, RTSP URL)

        Example:
            >>> realtime = RealtimeMode(config)
            >>> # 웹캠 사용
            >>> realtime.start_realtime_processing(0)
            >>> # RTSP 스트림 사용
            >>> realtime.start_realtime_processing('rtsp://192.168.1.100:554/stream')
            >>> # 비디오 파일 사용
            >>> realtime.start_realtime_processing('/path/to/video.mp4')
        """

    def process_frame_realtime(self, frame: np.ndarray) -> RealtimeResult:
        """
        실시간 프레임 처리

        Args:
            frame: 입력 프레임

        Returns:
            RealtimeResult: 실시간 처리 결과
        """

    def stop_processing(self) -> None:
        """실시간 처리 중지"""
```

RealtimeVisualizer

```python
class RealtimeVisualizer:
    """실시간 시각화기"""

    def __init__(self, config: VisualizationConfig):
        self.config = config
        self.overlay_mode = config.overlay_mode
        self.colors = self._setup_colors()

    def render_frame(self, frame: np.ndarray,
                     poses: List[TrackedPerson],
                     classification_result: ClassificationResult,
                     event_data: EventData) -> np.ndarray:
        """
        프레임 렌더링

        Args:
            frame: 원본 프레임
            poses: 추적된 포즈 리스트
            classification_result: 분류 결과
            event_data: 이벤트 데이터

        Returns:
            np.ndarray: 렌더링된 프레임

        Example:
            >>> visualizer = RealtimeVisualizer(config)
            >>> rendered_frame = visualizer.render_frame(
            ...     frame=frame,
            ...     poses=tracked_poses,
            ...     classification_result=result,
            ...     event_data=events
            ... )
            >>> cv2.imshow('Realtime Recognition', rendered_frame)
        """

    def draw_skeleton(self, frame: np.ndarray, person: TrackedPerson) -> np.ndarray:
        """스켈레톤 그리기"""

    def draw_bounding_box(self, frame: np.ndarray, person: TrackedPerson) -> np.ndarray:
        """바운딩 박스 그리기"""

    def draw_classification_info(self, frame: np.ndarray,
                                 result: ClassificationResult) -> np.ndarray:
        """분류 정보 표시"""

    def draw_event_overlay(self, frame: np.ndarray, event_data: EventData) -> np.ndarray:
        """이벤트 오버레이 표시"""
```

EventManager

```python
class EventManager:
    """이벤트 관리자"""

    def __init__(self, config: EventConfig):
        self.config = config
        self.active_events = {}
        self.event_history = []
        self.threshold_manager = ThresholdManager(config.thresholds)

    def process_classification_result(self, result: ClassificationResult,
                                      timestamp: float) -> EventData:
        """
        분류 결과 기반 이벤트 처리

        Args:
            result: 분류 결과
            timestamp: 타임스탬프

        Returns:
            EventData: 이벤트 데이터

        Example:
            >>> event_manager = EventManager(config)
            >>> event_data = event_manager.process_classification_result(
            ...     result=classification_result,
            ...     timestamp=time.time()
            ... )
            >>> if event_data.is_active:
            ...     print(f"Event detected: {event_data.event_type}")
        """

    def update_event_state(self, event_type: str, confidence: float,
                           timestamp: float) -> EventState:
        """이벤트 상태 업데이트"""

    def check_event_conditions(self, result: ClassificationResult) -> List[str]:
        """이벤트 조건 확인"""

    def get_active_events(self) -> List[EventData]:
        """활성 이벤트 목록 반환"""

    def log_event(self, event_data: EventData) -> None:
        """이벤트 로깅"""
```

3.3 Visualize Mode API

`VisualizeMode`

```python
class VisualizeMode(BaseMode):
    """시각화 모드"""

    def __init__(self, config: VisualizeConfig):
        super().__init__(config)
        self.visualizer = AdvancedVisualizer(config.visualization_config)
        self.video_encoder = VideoEncoder(config.encoding_config)

    def execute(self) -> ExecutionResult:
        """시각화 모드 실행"""

    def visualize_pkl_results(self, pkl_path: str, video_path: str,
                              output_path: str) -> VisualizationResult:
        """
        PKL 결과 시각화

        Args:
            pkl_path: PKL 결과 파일 경로
            video_path: 원본 비디오 파일 경로
            output_path: 출력 비디오 경로

        Returns:
            VisualizationResult: 시각화 결과

        Example:
            >>> visualize = VisualizeMode(config)
            >>> result = visualize.visualize_pkl_results(
            ...     pkl_path='results/video_analysis.pkl',
            ...     video_path='videos/input.mp4',
            ...     output_path='output/visualization.mp4'
            ... )
            >>> print(f"Created visualization: {result.output_path}")
        """

    def create_frame_sequence(self, pkl_data: Dict, video_frames: List[np.ndarray]) -> List[np.ndarray]:
        """프레임 시퀀스 생성"""

    def apply_visualization_style(self, style: str) -> None:
        """시각화 스타일 적용"""
```

4. Pipeline APIs

4.1 DualServicePipeline

```python
class DualServicePipeline(BasePipeline):
    """듀얼 서비스 통합 파이프라인"""

    def __init__(self, config: PipelineConfig):
        """
        듀얼 서비스 파이프라인 초기화

        Args:
            config: 파이프라인 설정

        Example:
            >>> config = PipelineConfig.from_yaml('pipeline_config.yaml')
            >>> pipeline = DualServicePipeline(config)
            >>> pipeline.initialize()
        """
        super().__init__(config)
        self.pose_estimator = RTMOONNXEstimator(config.pose_config)
        self.tracker = ByteTrackerWrapper(config.tracking_config)
        self.window_processor = SlidingWindowProcessor(config.window_config)

        # 서비스별 모듈
        self.services = {}
        if 'fight' in config.enabled_services:
            self.services['fight'] = FightService(config.fight_config)
        if 'falldown' in config.enabled_services:
            self.services['falldown'] = FalldownService(config.falldown_config)

    def initialize(self) -> None:
        """파이프라인 초기화"""

    async def process_frame(self, frame: np.ndarray) -> ProcessingResult:
        """
        프레임 처리

        Args:
            frame: 입력 프레임

        Returns:
            ProcessingResult: 처리 결과

        Example:
            >>> async def process_video():
            ...     cap = cv2.VideoCapture('video.mp4')
            ...     while True:
            ...         ret, frame = cap.read()
            ...         if not ret:
            ...             break
            ...         result = await pipeline.process_frame(frame)
            ...         print(f"Detected {len(result.poses)} persons")
        """

    def process_video_file(self, video_path: str) -> VideoProcessingResult:
```

```
        """비디오 파일 처리"""

    def get_performance_metrics(self) -> Dict[str, float]:
        """성능 메트릭 조회"""

    def reset_pipeline(self) -> None:
        """파이프라인 리셋"""
```

4.2 Service APIs

FightService

```
class FightService(BaseService):
    """폭력 행동 감지 서비스"""

    def __init__(self, config: FightConfig):
        """
        Fight 서비스 초기화

        Args:
            config: Fight 서비스 설정
        """
        super().__init__(config)
        self.scorer = MotionBasedScorer(config.scorer_config)
        self.classifier = STGCNClassifier(config.classifier_config)
        self.event_manager = FightEventManager(config.event_config)

    async def process_window(self, window_data: WindowData) -> ServiceResult:
        """
        윈도우 데이터 처리

        Args:
            window_data: 슬라이딩 윈도우 데이터

        Returns:
            ServiceResult: 서비스 처리 결과

        Example:
            >>> fight_service = FightService(config)
            >>> result = await fight_service.process_window(window_data)
            >>> if result.event_detected:
            ...     print(f"Fight detected with confidence: {result.confidence}")
        """

    def calculate_fight_features(self, poses: List[TrackedPerson]) -> np.ndarray:
        """폭력 행동 특징 계산"""

    def classify_action(self, features: np.ndarray) -> ClassificationResult:
        """행동 분류"""

    def detect_fight_events(self, classification_result: ClassificationResult) -> EventData:
        """폭력 이벤트 감지"""
```

FalldownService

```python
class FalldownService(BaseService):
    """낙상 감지 서비스"""

    def __init__(self, config: FalldownConfig):
        """
        Falldown 서비스 초기화

        Args:
            config: Falldown 서비스 설정
        """
        super().__init__(config)
        self.scorer = PoseBasedScorer(config.scorer_config)
        self.classifier = STGCNClassifier(config.classifier_config)
        self.event_manager = FalldownEventManager(config.event_config)

    async def process_window(self, window_data: WindowData) -> ServiceResult:
        """
        윈도우 데이터 처리

        Args:
            window_data: 슬라이딩 윈도우 데이터

        Returns:
            ServiceResult: 서비스 처리 결과

        Example:
            >>> falldown_service = FalldownService(config)
            >>> result = await falldown_service.process_window(window_data)
            >>> if result.event_detected:
            ...     print(f"Fall detected with confidence: {result.confidence}")
        """

    def calculate_falldown_features(self, poses: List[TrackedPerson]) -> np.ndarray:
        """낙상 특징 계산"""

    def analyze_body_orientation(self, pose: TrackedPerson) -> float:
        """신체 기울기 분석"""

    def calculate_height_ratio(self, pose: TrackedPerson) -> float:
        """높이 비율 계산"""

    def detect_fall_events(self, classification_result: ClassificationResult) -> EventData:
        """낙상 이벤트 감지"""
```

5. Configuration APIs

5.1 Configuration Management

`ConfigManager`

```python
class ConfigManager:
    """설정 관리자"""

    @staticmethod
    def load_config(config_path: str) -> Config:
        """
        설정 파일 로드

        Args:
            config_path: 설정 파일 경로

        Returns:
            Config: 설정 객체

        Example:
            >>> config = ConfigManager.load_config('config.yaml')
            >>> print(f"Mode: {config.mode}")
            >>> print(f"GPU enabled: {config.gpu.enabled}")
        """

    @staticmethod
    def validate_config(config: Config) -> ValidationResult:
        """설정 검증"""

    @staticmethod
    def merge_configs(base_config: Config, override_config: Config) -> Config:
        """설정 병합"""

    @staticmethod
    def save_config(config: Config, output_path: str) -> None:
        """설정 저장"""
```

Configuration Classes

```python
@dataclass
class Config:
    """메인 설정 클래스"""

    mode: str
    gpu: GPUConfig
    logging: LoggingConfig
    paths: PathsConfig
    performance: PerformanceConfig

    @classmethod
    def from_yaml(cls, yaml_path: str) -> 'Config':
        """YAML 파일에서 설정 로드"""

    def to_dict(self) -> Dict[str, Any]:
        """딕셔너리로 변환"""

    def validate(self) -> bool:
        """설정 유효성 검사"""

@dataclass
class GPUConfig:
    """GPU 설정"""

    enabled: bool = True
    device_ids: List[int] = field(default_factory=lambda: [0])
    memory_fraction: float = 0.8
    allow_growth: bool = True

@dataclass
class PipelineConfig:
    """파이프라인 설정"""

    pose_config: RTMOConfig
    tracking_config: TrackingConfig
    window_config: WindowConfig
    classification_config: ClassificationConfig
    enabled_services: List[str] = field(default_factory=lambda: ['fight', 'falldown'])
```

6. Utility APIs

6.1 Data Processing Utilities

`DataLoader`

```python
class DataLoader:
    """데이터 로더 유틸리티"""

    @staticmethod
    def load_pkl_file(pkl_path: str) -> Any:
        """
        PKL 파일 로드

        Args:
            pkl_path: PKL 파일 경로

        Returns:
            Any: 로드된 데이터

        Example:
            >>> data = DataLoader.load_pkl_file('stage1_poses.pkl')
            >>> print(f"Loaded {len(data)} frame poses")
        """

    @staticmethod
    def save_pkl_file(data: Any, pkl_path: str) -> None:
        """PKL 파일 저장"""

    @staticmethod
    def load_video_metadata(video_path: str) -> VideoMetadata:
        """비디오 메타데이터 로드"""

    @staticmethod
    def scan_video_directory(directory: str, extensions: List[str] = None) -> List[str]:
        """비디오 디렉토리 스캔"""
```

VideoProcessor

```python
class VideoProcessor:
    """비디오 처리 유틸리티"""

    def __init__(self, config: VideoConfig):
        self.config = config

    def load_video(self, video_path: str) -> cv2.VideoCapture:
        """
        비디오 로드

        Args:
            video_path: 비디오 파일 경로

        Returns:
            cv2.VideoCapture: 비디오 캡처 객체

        Example:
            >>> processor = VideoProcessor(config)
            >>> cap = processor.load_video('video.mp4')
            >>> fps = cap.get(cv2.CAP_PROP_FPS)
            >>> print(f"Video FPS: {fps}")
        """

    def extract_frames(self, video_path: str, frame_indices: List[int] = None) -> List[np.ndarray]:
        """프레임 추출"""

    def get_video_info(self, video_path: str) -> VideoInfo:
        """비디오 정보 조회"""

    def resize_frame(self, frame: np.ndarray, target_size: Tuple[int, int]) -> np.ndarray:
        """프레임 리사이즈"""
```

6.2 Performance Utilities

`PerformanceMonitor`

```python
class PerformanceMonitor:
    """성능 모니터링 유틸리티"""

    def __init__(self):
        self.metrics_buffer = deque(maxlen=1000)
        self.start_time = None

    def start_monitoring(self) -> None:
        """모니터링 시작"""

    def record_frame_time(self, processing_time: float) -> None:
        """
        프레임 처리 시간 기록

        Args:
            processing_time: 처리 시간 (초)

        Example:
            >>> monitor = PerformanceMonitor()
            >>> monitor.start_monitoring()
            >>> start = time.time()
            >>> # ... 프레임 처리 ...
            >>> monitor.record_frame_time(time.time() - start)
        """

    def get_current_fps(self) -> float:
        """현재 FPS 조회"""

    def get_average_latency(self) -> float:
        """평균 지연시간 조회"""

    def get_performance_summary(self) -> PerformanceMetrics:
        """성능 요약 조회"""
```

7. Error Handling

7.1 Exception Classes

```python
class RecognizerError(Exception):
    """Recognizer 기본 예외 클래스"""

    def __init__(self, message: str, error_code: str = None):
        super().__init__(message)
        self.error_code = error_code
        self.timestamp = time.time()

class VideoLoadError(RecognizerError):
    """비디오 로드 오류"""
    pass

class PoseEstimationError(RecognizerError):
    """포즈 추정 오류"""
    pass

class TrackingError(RecognizerError):
    """추적 오류"""
    pass

class ClassificationError(RecognizerError):
    """분류 오류"""
    pass

class ConfigurationError(RecognizerError):
    """설정 오류"""
    pass
```

7.2 Error Recovery

```python
class ErrorRecoveryManager:
    """오류 복구 관리자"""

    def __init__(self, config: ErrorRecoveryConfig):
        self.config = config
        self.retry_count = {}

    def handle_error(self, error: Exception, context: str) -> RecoveryAction:
        """
        오류 처리

        Args:
            error: 발생한 예외
            context: 오류 발생 컨텍스트

        Returns:
            RecoveryAction: 복구 액션

        Example:
            >>> recovery_manager = ErrorRecoveryManager(config)
            >>> try:
            ...     # ... 처리 로직 ...
            ... except PoseEstimationError as e:
            ...     action = recovery_manager.handle_error(e, 'pose_estimation')
            ...     if action == RecoveryAction.RETRY:
            ...         # 재시도 로직
        """

    def should_retry(self, error: Exception, context: str) -> bool:
        """재시도 여부 판단"""

    def reset_retry_count(self, context: str) -> None:
        """재시도 카운트 리셋"""
```

8. Performance Monitoring

8.1 Metrics Collection

```python
class MetricsCollector:
    """메트릭 수집기"""

    def __init__(self):
        self.metrics = {}
        self.collectors = []

    def collect_system_metrics(self) -> SystemMetrics:
        """
        시스템 메트릭 수집

        Returns:
            SystemMetrics: 시스템 메트릭

        Example:
            >>> collector = MetricsCollector()
            >>> metrics = collector.collect_system_metrics()
            >>> print(f"GPU Utilization: {metrics.gpu_utilization}%")
            >>> print(f"Memory Usage: {metrics.memory_usage}%")
            >>> print(f"CPU Usage: {metrics.cpu_usage}%")
        """

    def collect_inference_metrics(self) -> InferenceMetrics:
        """추론 메트릭 수집"""

    def export_metrics(self, format: str = 'json') -> str:
        """메트릭 내보내기"""

    def start_collection(self, interval: float = 1.0) -> None:
        """주기적 수집 시작"""
```

8.2 Performance Optimization

```python
class PerformanceOptimizer:
    """성능 최적화기"""

    def __init__(self, config: OptimizationConfig):
        self.config = config
        self.adaptive_settings = {}

    def optimize_for_realtime(self) -> OptimizationResult:
        """
        실시간 처리 최적화

        Returns:
            OptimizationResult: 최적화 결과

        Example:
            >>> optimizer = PerformanceOptimizer(config)
            >>> result = optimizer.optimize_for_realtime()
            >>> if result.optimized:
            ...     print(f"Optimized settings: {result.new_settings}")
        """

    def optimize_batch_size(self, current_performance: PerformanceMetrics) -> int:
        """배치 크기 최적화"""

    def optimize_model_precision(self) -> str:
        """모델 정밀도 최적화"""

    def suggest_hardware_settings(self) -> Dict[str, Any]:
        """하드웨어 설정 제안"""
```

사용 예제

완전한 파이프라인 실행 예제

```python
import asyncio
from recognizer.main import main
from recognizer.config import ConfigManager
from recognizer.modes import RealtimeMode, AnalysisMode
from recognizer.pipelines import DualServicePipeline

async def complete_pipeline_example():
    """완전한 파이프라인 실행 예제"""

    # 1. 설정 로드
    config = ConfigManager.load_config('config.yaml')

    # 2. 실시간 모드 실행
    realtime_mode = RealtimeMode(config)
    print("Starting realtime processing...")
    await realtime_mode.start_realtime_processing('rtsp://camera_url')

    # 3. 배치 분석 실행
    analysis_mode = AnalysisMode(config)
    video_files = ['/path/to/video1.mp4', '/path/to/video2.mp4']
    analysis_result = analysis_mode.analyze_video_batch(video_files)

    print(f"Analysis completed: {analysis_result.total_videos} videos processed")
    print(f"Average accuracy: {analysis_result.metrics.accuracy:.3f}")

# 실행
if __name__ == "__main__":
    asyncio.run(complete_pipeline_example())
```

Docker 환경에서의 실행 예제

```
import subprocess
import os

def run_in_docker():
    """Docker 환경에서 실행 예제"""

    # Docker 컨테이너에서 실시간 추론 실행
    cmd = [
        'docker', 'exec', 'mmlabs', 'bash', '-c',
        'cd /workspace/recognizer && python3 main.py --mode inference.realtime --log-level INFO'
    ]

    result = subprocess.run(cmd, capture_output=True, text=True)

    if result.returncode == 0:
        print("Realtime inference completed successfully")
    else:
        print(f"Error: {result.stderr}")

if __name__ == "__main__":
    run_in_docker()
```