```python
import numpy as np
import cv2
from typing import List, Dict, Any, Optional, Tuple
from dataclasses import dataclass, field
from src.mp_detect import mediapipe_pose_func, mediapipe_hand_func

@dataclass(frozen=True)
class SkeletonPoint:
    """단일 골격 포인트 (x, y, z) 및 visibility"""
    x: float
    y: float
    z: float
    visibility: Optional[float] = None # MediaPipe는 visibility도 제공

    def to_tuple(self) -> Tuple[float, float, float]:
        return (self.x, self.y, self.z)

class MediaPipePoseEstimator:
    def __init__(self, model_path: str):
        self.model_path = model_path
        # Note: model_path is kept for interface compatibility but not used by mp_detect
        pass

    def process_frame(self, image_np: np.ndarray) -> List[SkeletonPoint]:
        """
        전처리된 이미지 프레임(RGB numpy array)을 받아 골격 좌표를 추론합니다.
        Input: image_np (np.ndarray, RGB)
        Output: List[SkeletonPoint] (2D/3D Pose Landmarker Results)
        """
        # mp_detect.mediapipe_pose_func expects BGR image (it converts BGR->RGB internally)
        # So we convert our RGB input to BGR first.
        image_bgr = cv2.cvtColor(image_np, cv2.COLOR_RGB2BGR)

        # Returns np.array([all_x, all_y, all_z, all_vis])
        result_array = mediapipe_pose_func(image_bgr)

        all_landmarks: List[SkeletonPoint] = []

        # Check if we have valid results (shape should be (4, N) where N > 0)
        if result_array.ndim == 2 and result_array.shape[0] == 4 and result_array.shape[1] > 0:
            xs = result_array[0]
            ys = result_array[1]
            zs = result_array[2]
            vis = result_array[3]

            for i in range(len(xs)):
                all_landmarks.append(SkeletonPoint(
                    x=float(xs[i]),
                    y=float(ys[i]),
                    z=float(zs[i]),
                    visibility=float(vis[i])
                ))

        # Add Hand Landmarks
        # Returns np.array([all_x, all_y, all_z, all_side])
        hand_result_array = mediapipe_hand_func(image_bgr)

        if hand_result_array.ndim == 2 and hand_result_array.shape[0] == 4 and hand_result_array.shape[1] > 0:
            xs = hand_result_array[0]
            ys = hand_result_array[1]
            zs = hand_result_array[2]
            sides = hand_result_array[3] # 1.0 for Right, 0.0 for Left

            for i in range(len(xs)):
                all_landmarks.append(SkeletonPoint(
                    x=float(xs[i]),
                    y=float(ys[i]),
                    z=float(zs[i]),
                    visibility=float(sides[i])
                ))

        return all_landmarks
```