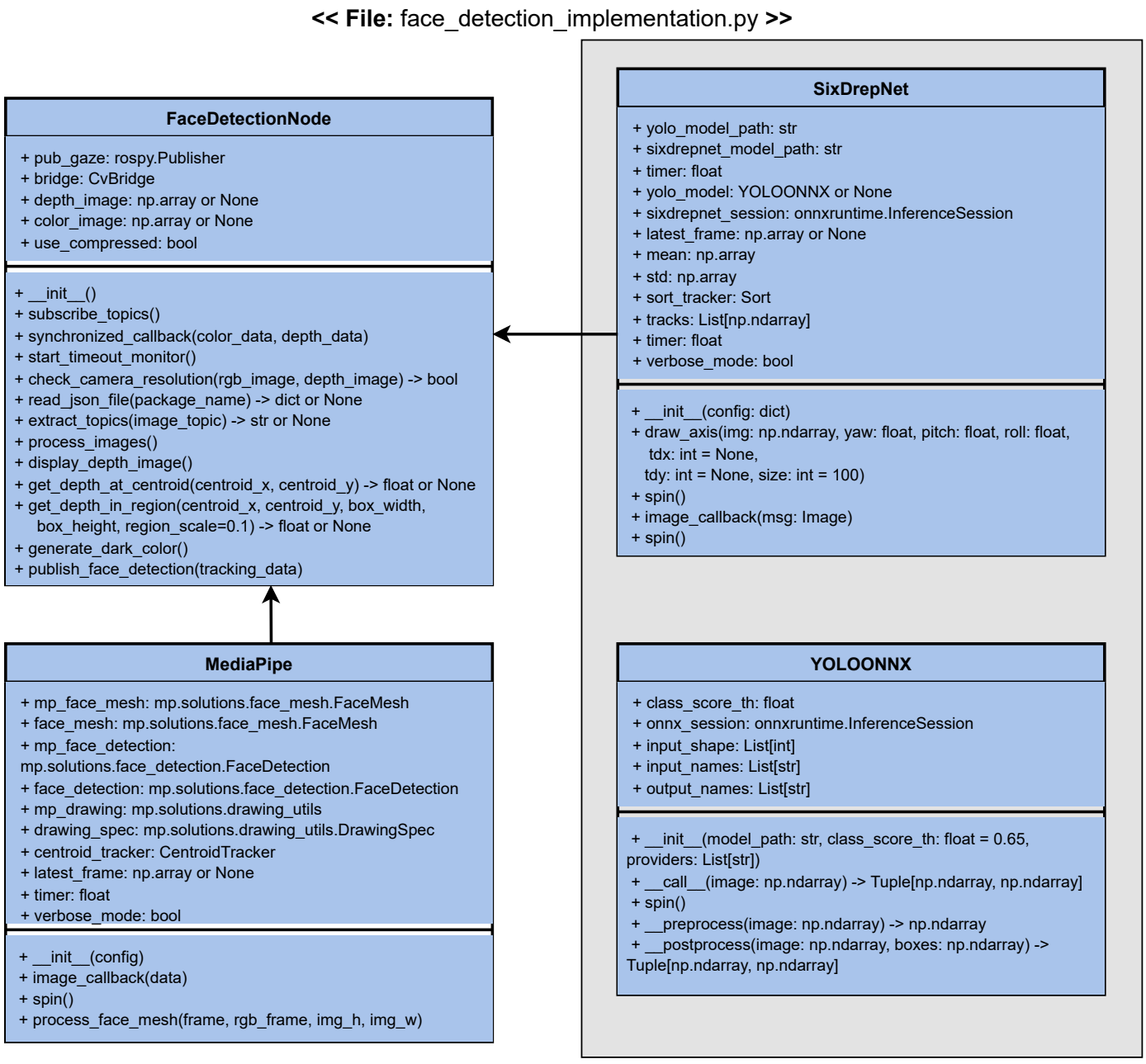
<< File: face_detection_implementation.py >>
FaceDetectionNode
+ pub_gaze: rospy.Publisher
+ bridge: CvBridge
+ depth_image: np.array or None
+ color_image: np.array or None
+ use_compressed: bool
+ __init__()
+ subscribe_topics()
+ synchronized_callback(color_data, depth_data)
+ start_timeout_monitor()
+ check_camera_resolution(rgb_image, depth_image) -> bool
+ read_json_file(package_name) -> dict or None
+ extract_topics(image_topic) -> str or None
+ process_images()
+ display_depth_image()
+ get_depth_at_centroid(centroid_x, centroid_y) -> float or None
+ get_depth_in_region(centroid_x, centroid_y, box_width, box_height, region_scale=0.1) -> float or None
+ generate_dark_color()
+ publish_face_detection(tracking_data)
SixDrepNet
+ yolo_model_path: str
+ sixdrepnet_model_path: str
+ timer: float
+ yolo_model: YOLOONNX or None
+ sixdrepnet_session: onnxruntime.InferenceSession
+ latest_frame: np.array or None
+ mean: np.array
+ std: np.array
+ sort_tracker: Sort
+ tracks: List[np.ndarray]
+ timer: float
+ verbose_mode: bool
+ __init__(config: dict)
+ draw_axis(img: np.ndarray, yaw: float, pitch: float, roll: float, tdx: int = None, tdy: int = None, size: int = 100)
+ spin()
+ image_callback(msg: Image)
+ spin()
MediaPipe
+ mp_face_mesh: mp.solutions.face_mesh.FaceMesh
+ face_mesh: mp.solutions.face_mesh.FaceMesh
+ mp_face_detection: mp.solutions.face_detection.FaceDetection
+ face_detection: mp.solutions.face_detection.FaceDetection
+ mp_drawing: mp.solutions.drawing_utils
+ drawing_spec: mp.solutions.drawing_utils.DrawingSpec
+ centroid_tracker: CentroidTracker
+ latest_frame: np.array or None
+ timer: float
+ verbose_mode: bool
+ __init__(config)
+ image_callback(data)
+ spin()
+ process_face_mesh(frame, rgb_frame, img_h, img_w)
YOLOONNX
+ class_score_th: float
+ onnx_session: onnxruntime.InferenceSession
+ input_shape: List[int]
+ input_names: List[str]
+ output_names: List[str]
+ __init__(model_path: str, class_score_th: float = 0.65, providers: List[str])
+ __call__(image: np.ndarray) -> Tuple[np.ndarray, np.ndarray]
+ spin()
+ __preprocess(image: np.ndarray) -> np.ndarray
+ __postprocess(image: np.ndarray, boxes: np.ndarray) -> Tuple[np.ndarray, np.ndarray]