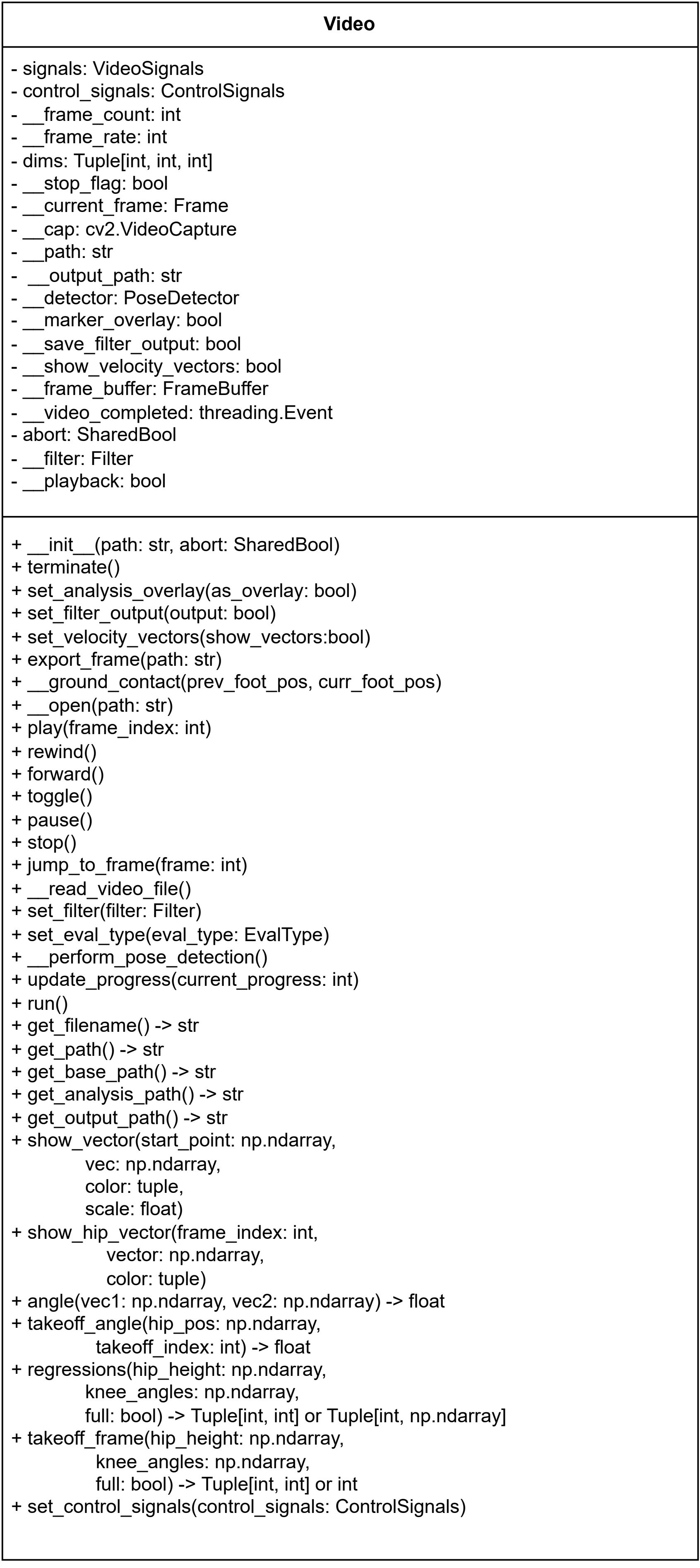

Video

```
- signals: VideoSignals
- control_signals: ControlSignals
- __frame_count: int
- __frame_rate: int
- dims: Tuple[int, int, int]
- __stop_flag: bool
- __current_frame: Frame
- __cap: cv2.VideoCapture
- __path: str
-  __output_path: str
- __detector: PoseDetector
- __marker_overlay: bool
- __save_filter_output: bool
- __show_velocity_vectors: bool
- __frame_buffer: FrameBuffer
- __video_completed: threading.Event
- abort: SharedBool
- __filter: Filter
- __playback: bool
```

```
+ __init__(path: str, abort: SharedBool)
+ terminate()
+ set_analysis_overlay(as_overlay: bool)
+ set_filter_output(output: bool)
+ set_velocity_vectors(show_vectors:bool)
+ export_frame(path: str)
+ __ground_contact(prev_foot_pos, curr_foot_pos)
+ __open(path: str)
+ play(frame_index: int)
+ rewind()
+ forward()
+ toggle()
+ pause()
+ stop()
+ jump_to_frame(frame: int)
+ __read_video_file()
+ set_filter(filter: Filter)
+ set_eval_type(eval_type: EvalType)
+ __perform_pose_detection()
+ update_progress(current_progress: int)
+ run()
+ get_filename() -> str
+ get_path() -> str
+ get_base_path() -> str
+ get_analysis_path() -> str
+ get_output_path() -> str
+ show_vector(start_point: np.ndarray,
            vec: np.ndarray,
            color: tuple,
            scale: float)
+ show_hip_vector(frame_index: int,
                vector: np.ndarray,
                color: tuple)
+ angle(vec1: np.ndarray, vec2: np.ndarray) -> float
+ takeoff_angle(hip_pos: np.ndarray,
              takeoff_index: int) -> float
+ regressions(hip_height: np.ndarray,
            knee_angles: np.ndarray,
            full: bool) -> Tuple[int, int] or Tuple[int, np.ndarray]
+ takeoff_frame(hip_height: np.ndarray,
              knee_angles: np.ndarray,
              full: bool) -> Tuple[int, int] or int
+ set_control_signals(control_signals: ControlSignals)
```