**Overall Workflow for video.tsx**

The Video component serves as the central hub for managing video playback, detecting chess pieces, and updating the chessboard state. It integrates with AI models, Redux, and other utilities in the project to provide a seamless real-time chess detection experience. Below is a step-by-step breakdown.

**1. Import Statements**

The file starts with importing essential libraries, components, and utilities:

import { findPieces } from "../../utils/findPieces"; *// Function to detect chess pieces in video frames*

import { useEffect, useRef } from "react"; *// React hooks for managing side effects and references*

import { CORNER\_KEYS, MARKER\_DIAMETER, MARKER\_RADIUS, MEDIA\_ASPECT\_RATIO, MEDIA\_CONSTRAINTS } from "../../utils/constants"; *// Constants for UI and video settings*

import { Corners } from "."; *// Component to display detected chessboard corners*

import { useWindowWidth, useWindowHeight } from '@react-hook/window-size'; *// Hooks for dynamically getting window size*

import { useDispatch } from 'react-redux'; *// Hook for dispatching Redux actions*

import { cornersSet } from "../../slices/cornersSlice"; *// Redux action to update chessboard corner state*

import { getMarkerXY, getXY } from "../../utils/detect"; *// Utilities for handling marker positions and coordinates*

import { CornersPayload, Game, Mode, MovesPair, SetBoolean, SetStringArray } from "../../types"; *// Type definitions for the app*

import { gameSelect, makeBoard } from "../../slices/gameSlice"; *// Redux selectors and functions for managing chess game state*

import { getMovesPairs } from "../../utils/moves"; *// Utility to calculate valid moves based on chessboard state*

import { Chess } from "chess.js"; *// Library for chess game logic*

**Key Connections:**

1. findPieces: Handles piece detection and chessboard state updates.

2. **Redux Actions**: Updates chessboard corners and game state via cornersSet and gameSelect.

3. **Chess.js**: Manages chessboard logic and move validation.

**2. Component Setup**

The Video component initializes essential references, Redux dispatchers, and hooks.

const game: Game = gameSelect(); *// Select the current chess game state from Redux*

const displayRef = useRef<any>(null); *// Reference for the display container*

const boardRef = useRef<Chess>(makeBoard(game)); *// Reference to the Chess.js board instance*

const movesPairsRef = useRef<MovesPair[]>(getMovesPairs(boardRef.current)); *// Valid move pairs*

const lastMoveRef = useRef<string>(game.lastMove); *// Last move reference*

const moveTextRef = useRef<string>(""); *// Move text reference*

const windowWidth = useWindowWidth(); *// Current window width*

const windowHeight = useWindowHeight(); *// Current window height*

const dispatch = useDispatch(); *// Redux dispatch function*

**3. Handling Game State Updates**

Whenever the game state changes, the chessboard and valid moves are updated.

useEffect(() => {

  const board = makeBoard(game); *// Create a new board based on the game state*

  moveTextRef.current = getMoveText(board); *// Convert moves into text format*

  if (game.greedy) {

    board.undo(); *// Undo the last move in "greedy" mode*

  } else {

    movesPairsRef.current = getMovesPairs(board); *// Update move pairs*

  }

  boardRef.current = board; *// Update the board reference*

  lastMoveRef.current = game.lastMove; *// Update the last move*

}, [game]);

**4. Video Setup and Dimensions**

The component ensures proper initialization of the video source and canvas dimensions.

const setupWebcam = async () => {

  const stream = await navigator.mediaDevices.getUserMedia(MEDIA\_CONSTRAINTS); *// Request webcam access*

  if (videoRef.current !== null) {

    videoRef.current.srcObject = stream; *// Attach the stream to the video element*

  }

  return stream; *// Return the stream for cleanup*

};

const updateWidthHeight = () => {

  let height = ((windowWidth - sidebarRef.current.offsetWidth - MARKER\_DIAMETER)

  / MEDIA\_ASPECT\_RATIO) + MARKER\_DIAMETER;

  if (height > windowHeight) height = windowHeight; *// Adjust for window height*

  displayRef.current.style.height = `${height}px`; *// Apply calculated dimensions*

};

**5. Real-Time Detection and Updates**

findPieces integrates AI-based detection into the video feed.

useEffect(() => {

  updateWidthHeight(); *// Initial setup*

  let streamPromise: any = null;

  if (mode !== "upload") {

    streamPromise = await setupWebcam(); *// Initialize webcam*

  }

  findPieces(piecesModelRef, videoRef, canvasRef, playingRef, setText, dispatch,

    cornersRef, boardRef, movesPairsRef, lastMoveRef, moveTextRef, mode); *// Start piece detection*

  return () => {

    if (streamPromise) {

      streamPromise.getTracks().forEach((track: any) => track.stop()); *// Cleanup*

    }

  };

}, []);

**Key Function:**

• findPieces: Detects chess pieces and updates the board state.

**6. Event Handling and Rendering**

Handles video events and renders the updated chessboard state.

const onLoadedMetadata = () => {

*// Adjust video settings when metadata is loaded*

};

const onEnded = () => {

  setPlaying(false); *// Stop playback*

};

return (

  <div className="d-flex align-top justify-content-center">

    <div ref={displayRef}>

      <video ref={videoRef} onLoadedMetadata={onLoadedMetadata} onEnded={onEnded} />

      <canvas ref={canvasRef} />

    </div>

    <Corners /> {/\* Render detected chessboard corners \*/}

  </div>

);

**Key Relationships:**

1. gameSlice: Syncs game state with detected pieces.

2. findPieces: Bridges AI-based detection with real-time rendering.

3. Corners: Displays chessboard markers on the canvas.

This workflow provides a comprehensive view of how video.tsx works, from video initialization to AI-based piece detection and rendering. Let me know if you need further clarification or details!