app.R

tuq55540

2025-06-05

```r
### BehaviorDEPOT-Free v1.5.5 Full Lab Release (Mac version)

# --- Load Required Packages ---
library(shiny)
library(hdf5r)
library(magick)
```

```
## Linking to ImageMagick 6.9.12.93
## Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```r
library(shinyjs)
```

```
##
## Attaching package: 'shinyjs'

## The following object is masked from 'package:shiny':
##
##     runExample

## The following objects are masked from 'package:methods':
##
##     removeClass, show
```

```r
library(sp)
library(tidyr)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:hdf5r':
##
##     flatten_df
```

```r
options(shiny.maxRequestSize = 2000 * 1024^2)

# Color palette for ROI drawing
roi_colors <- c("red", "blue", "green", "purple", "orange", "cyan", "magenta", "yellow", "brown", "pink

# --- Extract still frame using ffmpeg (Mac-safe) ---
extract_frame <- function(video_path, time_sec) {
  ffmpeg_path <- "/opt/homebrew/bin/ffmpeg"
  tmpfile <- tempfile(fileext = ".png")
  cmd <- sprintf('"%s" -ss %.2f -i "%s" -vframes 1 -update 1 "%s"',
                 ffmpeg_path, time_sec, video_path, tmpfile)
  system(cmd)
  tmpfile
}

# --- Extract video FPS using ffprobe ---
extract_fps <- function(video_path) {
  ffprobe_path <- "/opt/homebrew/bin/ffprobe"
  cmd <- sprintf('"%s" -v error -select_streams v:0 -show_entries stream=r_frame_rate -of default=nokey
                 ffprobe_path, video_path)
  fps_raw <- system(cmd, intern = TRUE)
  fps_raw <- trimws(fps_raw)
  if (grepl("/", fps_raw)) {
    parts <- unlist(strsplit(fps_raw, "/"))
    fps <- as.numeric(parts[1]) / as.numeric(parts[2])
  } else {
    fps <- as.numeric(fps_raw)
  }
  if (is.na(fps) || fps <= 0) stop("Failed to parse valid FPS.")
  return(fps)
}

# --- Merge tracks across identities for given node ---
merge_tracks <- function(h5_path, node) {
  h5 <- H5File$new(h5_path, mode = "r")
  tracks <- h5[["tracks"]]$read()
  node_names <- sapply(h5[["node_names"]]$read(), function(x) gsub("\\x00", "", x))
  node_idx <- which(node_names == node)
  h5$close_all()
  n_frames <- dim(tracks)[1]
  n_tracks <- dim(tracks)[4]
  merged <- matrix(NA, nrow = n_frames, ncol = 2)
  for (f in 1:n_frames) {
    for (t in 1:n_tracks) {
      coords <- tracks[f, node_idx, , t]
      if (!any(is.na(coords))) {
        merged[f, ] <- coords
        break
      }
    }
  }
  return(merged)
}

# --- Compute ROI occupancy for one node ---
compute_roi_occupancy <- function(coords, roi_def, fps) {
  results <- data.frame(Frame = 1:nrow(coords))
  for (roi_name in names(roi_def)) {
    poly <- roi_def[[roi_name]]
    inside <- point.in.polygon(coords[,1], coords[,2], poly[,1], poly[,2]) > 0
    results[[roi_name]] <- inside
  }
  summary <- results %>%
    summarise(across(-Frame, ~ sum(.x, na.rm = TRUE))) %>%
    pivot_longer(cols = everything(), names_to = "ROI", values_to = "Frames") %>%
    mutate(Time_sec = Frames / fps)
  return(summary)
}

### --- UI LAYOUT ---

ui <- fluidPage(
  useShinyjs(),
  titlePanel("BehaviorDEPOT-Free v1.5.5 Full Lab Release"),
  sidebarLayout(
    sidebarPanel(
      fileInput("video", "Upload Video (MP4)", accept = ".mp4"),
      numericInput("frame_sec", "Extract frame at (sec):", value = 10, min = 0),
      actionButton("extract", "Extract Frame for ROI"),
      hr(),
      fileInput("h5file", "Upload SLEAP .h5 File", accept = ".h5"),
      textInput("roi_label", "ROI Name:", placeholder = "Enter ROI name here"),
      actionButton("save_roi", "Save ROI"),
      downloadButton("downloadROIs", "Download ROI Definitions"),
      hr(),
      actionButton("analyze", "Run Analysis", disabled = TRUE),
      downloadButton("downloadResults", "Download Results CSV"),
      downloadButton("downloadTimePlot", "Save Time Plot (JPG)"),
      downloadButton("downloadHeatmap", "Save Heatmap (JPG)"),
      downloadButton("downloadMissing", "Save Missing Plot (JPG)"),
      downloadButton("downloadRaster", "Save Raster Plot (JPG)"),
      selectInput("heatmapNode", "Heatmap Node:", choices = c("nose"), selected = "nose")
    ),
    mainPanel(
      plotOutput("frameplot", click = "plot_click"),
      plotOutput("timeplot"),
      plotOutput("heatmap"),
      plotOutput("missingplot"),
      plotOutput("rasterplot")
    )
  )
)
### --- SERVER LOGIC ---

server <- function(input, output, session) {

  frame_path <- reactiveVal()
  roi_points <- reactiveVal(data.frame(x = numeric(0), y = numeric(0)))
  roi_list <- reactiveVal(data.frame(Name = character(), x = numeric(), y = numeric(), stringsAsFactors
  analysis_results <- reactiveVal()
  node_list <- reactiveVal()

  observeEvent(input$extract, {
    req(input$video)
    path <- input$video$datapath
    frame_path(extract_frame(path, input$frame_sec))

    output$frameplot <- renderPlot({
      img <- image_read(frame_path())
      plot(img)
      unique_rois <- unique(roi_list()$Name)
      for (i in seq_along(unique_rois)) {
        pts <- roi_list()[roi_list()$Name == unique_rois[i], c("x","y")]
        polygon(c(pts$x, pts$x[1]), c(pts$y, pts$y[1]), col = rgb(1,0,0,0.3), border = roi_colors[i %%
        text(mean(pts$x), mean(pts$y), labels = unique_rois[i])
      }
      pts <- roi_points()
      if (nrow(pts) > 0) {
```

```
      polygon(c(pts$x, pts$x[1]), c(pts$y, pts$y[1]), col = rgb(1,0,0,0.5), border = "red")
      points(pts$x, pts$y, pch=16, col="red")
    }
  })
  shinyjs::enable("analyze")
})

observeEvent(input$plot_click, {
  pts <- roi_points()
  pts <- rbind(pts, data.frame(x=input$plot_click$x, y=input$plot_click$y))
  roi_points(pts)
  session$sendCustomMessage(type = "replot", message = list())
})

observeEvent(input$save_roi, {
  req(input$roi_label)
  req(nrow(roi_points()) > 0)
  current <- roi_points()
  label <- input$roi_label
  for (i in 1:nrow(current)) {
    roi_list(rbind(roi_list(), data.frame(Name = label, x = current[i,"x"], y = current[i,"y"], strin
  }
  roi_points(data.frame(x = numeric(0), y = numeric(0)))
  updateTextInput(session, "roi_label", value = "")
})

observeEvent(input$analyze, {
  req(input$h5file)
  req(input$video)

  h5 <- H5File$new(input$h5file$datapath, mode = "r")
  tracks <- h5[["tracks"]]$read()
  node_names <- sapply(h5[["node_names"]]$read(), function(x) gsub("\\x00", "", x))
  h5$close_all()
  node_list(node_names)

  updateSelectInput(session, "heatmapNode", choices = node_names, selected = node_names[1])

  fps <- extract_fps(input$video$datapath)
  roi_split <- split(roi_list()[,c("x","y")], roi_list()$Name)
  roi_split <- lapply(roi_split, function(df) as.matrix(df))

  all_results <- list()

  for (node in node_names) {
    coords <- merge_tracks(input$h5file$datapath, node)
    res <- compute_roi_occupancy(coords, roi_split, fps)
    res$Node <- node
    all_results[[node]] <- res
  }

  full_results <- bind_rows(all_results)
  analysis_results(full_results)
```

```r
output$timeplot <- renderPlot({
  ggplot(full_results, aes(x=ROI, y=Time_sec, fill=Node)) +
    geom_col(position="dodge") + theme_minimal() + ylab("Time (s)") + ggtitle("ROI Occupancy by Node
})

output$heatmap <- renderPlot({
  selected_node <- input$heatmapNode
  coords <- merge_tracks(input$h5file$datapath, selected_node)
  valid <- coords[complete.cases(coords),]
  if (nrow(valid) > 10) {
    kde <- MASS::kde2d(valid[,1], valid[,2], n=100)
    image(kde, col=viridis(50), main=paste("Heatmap:", selected_node))
  } else {
    plot(1, type="n", main="Insufficient data for heatmap")
  }
})

output$missingplot <- renderPlot({
  h5 <- H5File$new(input$h5file$datapath, mode = "r")
  tracks <- h5[["tracks"]]$read()
  h5$close_all()
  n_frames <- dim(tracks)[1]
  n_nodes <- dim(tracks)[2]
  missing_perc <- sapply(1:n_nodes, function(i) {
    valid <- apply(tracks[,i,,], 1, function(x) any(!is.na(x)))
    100 * sum(!valid) / n_frames
  })
  barplot(missing_perc, names.arg=node_names, main="Missing % by Node", ylab="% Missing")
})

output$rasterplot <- renderPlot({
  h5 <- H5File$new(input$h5file$datapath, mode = "r")
  tracks <- h5[["tracks"]]$read()
  node_names <- sapply(h5[["node_names"]]$read(), function(x) gsub("\\x00", "", x))
  h5$close_all()

  n_frames <- dim(tracks)[1]
  n_nodes <- dim(tracks)[2]
  n_tracks <- dim(tracks)[4]

  roi_split <- split(roi_list()[,c("x","y")], roi_list()$Name)
  roi_split <- lapply(roi_split, function(df) as.matrix(df))

  all_data <- list()

  for (node_idx in 1:n_nodes) {
    node <- node_names[node_idx]
    merged <- matrix(NA, nrow = n_frames, ncol = 2)
    for (f in 1:n_frames) {
      for (t in 1:n_tracks) {
        coords <- tracks[f, node_idx, , t]
        if (!any(is.na(coords))) {
          merged[f,] <- coords
```

```
              break
            }
          }
        }
        roi_assignments <- rep(NA, n_frames)
        for (roi_name in names(roi_split)) {
          poly <- roi_split[[roi_name]]
          inside <- point.in.polygon(merged[,1], merged[,2], poly[,1], poly[,2]) > 0
          roi_assignments[inside] <- roi_name
        }
        node_df <- data.frame(Frame = 1:n_frames, ROI = factor(roi_assignments, levels = names(roi_spli
        all_data[[node]] <- node_df
      }

      raster_df <- bind_rows(all_data)

      ggplot(raster_df, aes(x = Frame, y = ROI, fill = ROI)) +
        geom_tile(na.rm=FALSE) +
        facet_wrap(~Node, ncol=1, scales="free_y") +
        scale_fill_manual(values = roi_colors[1:length(unique(raster_df$ROI))], na.value = "black") +
        theme_minimal() + ggtitle("ROI Raster by Node") +
        theme(axis.text.y = element_text(size = 10), strip.text = element_text(size = 12))
    })
  })

  ### DOWNLOAD HANDLERS

  output$downloadResults <- downloadHandler(
    filename = function() {"results.csv"},
    content = function(file) { write.csv(analysis_results(), file, row.names=FALSE) }
  )

  output$downloadROIs <- downloadHandler(
    filename = function() {"roi_definitions.csv"},
    content = function(file) { write.csv(roi_list(), file, row.names=FALSE) }
  )

  output$downloadTimePlot <- downloadHandler(
    filename = function() {"timeplot.jpg"},
    content = function(file) {
      jpeg(file, width=1600, height=1200, quality=100)
      print(output$timeplot())
      dev.off()
    }
  )

  output$downloadHeatmap <- downloadHandler(
    filename = function() {"heatmap.jpg"},
    content = function(file) {
      jpeg(file, width=1600, height=1200, quality=100)
      print(output$heatmap())
      dev.off()
    }
  )

  output$downloadMissing <- downloadHandler(
    filename = function() {"missing.jpg"},
    content = function(file) {
      jpeg(file, width=1600, height=1200, quality=100)
      print(output$missingplot())
      dev.off()
    }
  )

  output$downloadRaster <- downloadHandler(
    filename = function() {"raster.jpg"},
    content = function(file) {
      jpeg(file, width=1600, height=1400, quality=100)
      print(output$rasterplot())
      dev.off()
    }
  )
}

shinyApp(ui, server)
```