```
import cv2
import mediapipe as mp
from math import hypot
import screen_brightness_control as sbc
import numpy as np

cap = cv2.VideoCapture(0)  # Capture video from webcam
mpHands = mp.solutions.hands
hands = mpHands.Hands()  # Initialize the hand-tracking module
mpDraw = mp.solutions.drawing_utils

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)  # Process the frame for hand landmarks

    lmList = []
    if results.multi_hand_landmarks:
        for handLandmark in results.multi_hand_landmarks:
            for id, lm in enumerate(handLandmark.landmark):
                h, w, _ = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                lmList.append([id, cx, cy])  # Collect landmark positions
            mpDraw.draw_landmarks(img, handLandmark, mpHands.HAND_CONNECTIONS)

        if lmList != []:
            # Get coordinates for thumb tip and index finger tip
            x1, y1 = lmList[4][1], lmList[4][2]
            x2, y2 = lmList[8][1], lmList[8][2]

            # Draw circles on the thumb and index finger tips
            cv2.circle(img, (x1, y1), 4, (255, 0, 0), cv2.FILLED)
            cv2.circle(img, (x2, y2), 4, (255, 0, 0), cv2.FILLED)

            # Draw a line between thumb and index finger
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 0), 3)

            # Calculate the length between thumb and index finger
            length = hypot(x2 - x1, y2 - y1)

            # Interpolate brightness value based on finger distance
            bright = np.interp(length, [15, 220], [0, 100])
            print(bright, length)

            # Set the screen brightness
            sbc.set_brightness(int(bright))

    cv2.imshow('Image', img)  # Display the processed frame
    if cv2.waitKey(1) & 0xFF == ord('q'):  # Break loop on 'q' key press
        break

cap.release()
cv2.destroyAllWindows()
```