Evaluación de la orientación del rostro en ambiente de trabajo con múltiples pantallas. Un enfoque práctico

Nelson Garrido,
Docentes: Carlos Neil, Silvia Poncio, Nicolás Battaglia

Universidad Abierta Interamericana, Facultad de Tecnología Informática,
Centro de Altos Estudios en Tecnología Informática, Rosario, Argentina
nelsonariel.garrido@alumnos.uai.edu.ar
{Carlos.Neil,Silvia.Poncio,Nicolas.Battaglia}@uai.edu.ar

Resumen. Al utilizar una computadora hoy en día, es frecuente el uso de múltiples pantallas para mejorar la productividad. Sin embargo, esto tiene sus desafíos. El mayor espacio de visualización aumenta la distancia que el cursor debe recorrer, lo que puede reducir la eficiencia y requerir de mayor concentración. Además, cambiar de una pantalla a otra puede generar confusión y distracción al perder la ubicación del puntero. Para abordar este problema, se desarrolló un prototipo que utiliza un modelo de aprendizaje automático preentrenado para detectar la orientación de la mirada del usuario y así determinar hacia cuál de las pantallas está dirigida su atención. Este demostró una alta precisión y bajo costo computacional, lo que sugiere que la solución podría mejorar el rendimiento en estos casos. Este trabajo presenta un análisis detallado de los inconvenientes que surgen al utilizar esta configuración, junto con una solución práctica que puede ayudar a mitigar estos problemas.

Keywords: Multi-monitor, Técnica de interacción, Detección de rostro

1 Introducción

La interacción humano-computadora (HCI) ha sido un campo de investigación activo durante varias décadas. Su objetivo es mejorar la eficiencia y la experiencia del usuario en la utilización de los sistemas informáticos. En los últimos tiempos, se ha explorado la optimización de la interacción mediante la creación de entornos con pantallas más grandes o la incorporación de múltiples pantallas.

Como es sabido existe una clara tendencia en la industria hacia monitores más grandes, ya sea con una sola superficie de visualización o en configuraciones de varios monitores [1]. Los entornos de múltiples pantallas se han vuelto comunes tanto en el hogar como en la oficina [2], por otro lado, en un ambiente estándar de trabajo o estudio es común que se utilice un portátil y un monitor (o más) que amplían la pantalla del ordenador. Como se plantea en [3] un número creciente de usuarios, a partir de 2005, han empezado a aprovechar la opción de utilizar múltiples monitores, capacidad que ha sido compatible durante algún tiempo en varios sistemas operativos y potenciada por el avance de la tecnología de tarjetas gráficas.

En contraste, como se expresa en [4] hasta resulta contraintuitivo pensar que el uso de varias pantallas puede ayudar a mantener el foco de atención en un trabajo en vez de generar distracción, pero su uso facilita el hecho de que múltiples tareas estén disponibles a primera vista en las pantallas. Además, ha quedado demostrado en diferentes estudios citados en [3] que este tipo de ambiente brinda una experiencia más inmersiva al realizar distintas tareas. Claramente, la coordinación del trabajo en múltiples pantallas es una estrategia que puede mejorar potencialmente el desempeño al enfocar la atención y reducir la carga de trabajo mental [4].

Las ventajas del uso de múltiples pantallas es un hecho, no obstante, al incrementar el espacio de trabajo también surgen algunas cuestiones que pueden ser analizadas. Por ejemplo, las mayores distancias a menudo afectan la forma en que los usuarios manejan el mouse [5]. En [1] se identifican cuatro cuestiones que pueden inhibir la productividad del usuario: mantener el seguimiento del cursor, acceso distal a ventanas e iconos, tratar con biseles, gestión de ventanas y gestión de tareas. Por otro lado, en [3] a los puntos mencionados le suma el problema de la falta de aprovechamiento de los periféricos de la máquina.

De los posibles inconvenientes que pueden surgir de la utilización de múltiples pantallas nombrados anteriormente, se pretende analizar y brindar una visión distinta a la cuestión del seguimiento del cursor. Precisamente lo que se pretende estudiar es la pérdida del seguimiento al producirse un cruce entre las pantallas. Un mecanismo para mover el cursor de una pantalla a otra es fundamental para las interfaces multi-pantalla, el rendimiento en el uso del ordenador puede ser mejorado drásticamente si elegimos técnicas de movimiento óptimas [6]. En el trabajo mencionado anteriormente, se dividen las técnicas de cruce de un monitor a otro en cuatro grupos: Stitching, Cursor Warping, Mouse Ether, and Portals (Tabla 1).

Tabla 1. Técnicas de movimiento del cursor entre pantallas (adaptado de [6])**.**

Stitching	Es la forma predeterminada de lidiar con el espacio entre pantallas. El espacio entre las pantallas simplemente se ignora. Este aún forma parte del espacio visual del usuario, lo que provoca una falta de coincidencia entre la acción realizada y la percepción.
Cursor Warping	Es una variante de Stitching en la que el cursor salta desde cualquier ubicación en una pantalla a otra ubicación en una pantalla diferente.
Mouse Ether	Fue diseñado para resolver algunos problemas de las implementaciones actuales de Stitching. Tiene en cuenta el tamaño real de las pantallas y el espacio entre ellas. Permite a los usuarios "tomar atajos". Así algunos movimientos requeridos se vuelven rectilíneos y, por lo tanto, más cortos que sus contrapartes de Stitching.
Portals	Es una forma completamente diferente de brindar acceso a otras pantallas. Se generan “portales” que replican el contenido de una pantalla diferente y permiten la manipulación del mismo, sin tener que desplazar el cursor al original.

En esta línea, [6] cita algunas investigaciones que estaban aplicando el uso del seguimiento de la cabeza y los ojos para el manejo del cambio de pantalla del cursor. Se plantea que, aunque estos enfoques son prometedores, requieren equipamientos que aún son caros e incomodos. Basándonos en esto, la idea del trabajo presentado es que a través de la cámara del ordenador o una externa se capture en tiempo real el rostro del operador y sus movimientos. Utilizando el modelo preentrando de *FaceMesh* que trabaja bajo *TensorFlow* se detectan las diferentes regiones del rosto que son utilizadas para calcular hacia cuál de las pantallas el operador está mirando.

Como paso previo a la utilización del sistema, se requiere de una configuración, tal como el ingreso de la cantidad de pantallas que se están utilizando y las áreas de visión que serán representativas para el paso de una pantalla a la otra. Todas las herramientas utilizadas aquí funcionan en *JavaScript*, por lo tanto, el modelo puede ser ejecutado en un browser sin la necesidad de servidor o soporte de *back-end*.

El objetivo de este trabajo es plantear y abordar los posibles inconvenientes que surgen al desarrollar un modelo de este tipo, con el fin de proponer una solución efectiva. Para poder hacer un análisis empírico se creó un prototipo cuya finalidad es detectar hacia cuál de las pantallas el operador está dirigiendo su mirada.

Las utilidades derivadas incluyen mover el mouse de una pantalla a otra según la dirección de la mirada, dar el foco a ciertas ventanas y detener la reproducción de un video cuando el usuario no está viendo esa pantalla, entre otras posibilidades, que se escapan al objetivo de este estudio.

Este artículo contiene cuatro secciones principales: la Sección 2 aborda el "Seguimiento de la mirada" y describe el enfoque y metodología utilizada para detectar la orientación del rostro del usuario. En la Sección 3, se presenta la "Prueba práctica en prototipo" con detalles sobre los resultados experimentales obtenidos y la evaluación del rendimiento del prototipo desarrollado. La Sección 4, titulada "Trabajos Relacionados", revisa otros estudios e investigaciones relacionadas con el tema. Por último, en la Sección 5 se exponen las "Conclusiones", donde se discuten los puntos clave del trabajo y se hacen comentarios sobre posibles desarrollos futuros.

2 Seguimiento de la mirada

Para este trabajo, se considera que la detección de la orientación de la mirada del usuario está más ligada al análisis de la posición de la cabeza (human posse estimation, HPE) que al movimiento de los ojos. En un contexto de visión por computadora, la estimación de la postura de la cabeza es el proceso de inferir o predecir la orientación partir de imágenes digitales [7] [8]. Esta estimación puede ser evaluada basándose en el ángulo de Euler de la cabeza, estudio que se centra en los tres ejes [9], para este fin se describen las rotaciones intrínsecas alrededor de X;Y;Z [8], dando como resultado los tres grados de libertad de movimiento con los que cuenta la cabeza humana (Pitch, Yaw, Roll; Fig. 1). Al utilizar esta técnica de análisis, la precisión está limitada por la exactitud del modelo facial y los puntos de referencia detectados [10], se podría decir que la clave en este proceso radica en la selección de una herramienta de detección de imágenes que sea confiable para este tipo de proyectos.

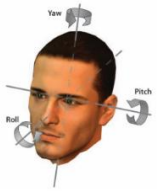

Fig. 2. Tres grados de libertad de la cabeza humana. Tomado de [7]

Por lo tanto, en un ambiente de trabajo con multi-monitor estas estimaciones permitirían el proceso de determinar hacia cuál pantalla se dirige la atención. En [11] se plantea que para realizar HPE, se pueden aplicar técnicas de preprocesamiento para encontrar la región de la cabeza o para detectar algún punto clave en la cara. Y [11] añade que estas se pueden dividir en tres grupos principales: detección de rostros; detección de puntos de referencia; modelado de cabezas en 3D.

Tal lo establecido en [12], el método tradicional de cálculo de la postura de la cabeza es estimar los puntos clave de un rostro objetivo para poder modelar la cabeza. Esta idea puede ser implementada utilizando herramientas de procesamiento de imágenes que trabajan con inteligencia artificial. Este trabajo se basa en este enfoque, se localizan áreas del rostro que son relevantes para el cálculo de la HPE.

Para crear un modelo de estudio, se optó por el *framework FaceMesh* desarrollado por *TensorFlow*. El reconocimiento de HPE se logra mediante el uso de una sola cámara, que captura un *streaming* del rostro y permite segmentarlo en sus componentes, como los ojos, la boca y la nariz, para su análisis. Todo esto se realiza considerando los movimientos del rostro, sin que la calidad de detección se vea afectada por los gestos realizados. Esta herramienta se basa en un modelo preentrenado que detecta el rostro del operador con una latencia muy baja, lo que permite trabajar en tiempo real y obtener como resultado una predicción en formato JSON. Dicho formato contiene las diferentes partes del rostro detectadas y su posición relativa frente a la cámara.

Ejemplo del json que se obtiene usando el método Predict.

```
  [
      {
          boundingBox: {
              topLeft: [{value, value}],
              bottomRight: [{value, value}]
          },
          annotations: [
                  {name: "rightEyel", x: value, y: value,
z: value}
                  ....
          ],
      }
  ]
```

Una vez que el *streaming* está inicializado, se utiliza un método proporcionado por la herramienta para obtener la lista de puntos clave que describe la figura del rostro en pantalla. Según la documentación oficial de ML5, el array devuelto consta de dos listas

principales: *BoundingBox*, que es el contorno del rostro, y *Annotations*, que es la parte relevante para este trabajo, ya que contiene cada uno de los puntos de referencia que componen las áreas del rostro detectado. Cada entrada de la lista incluye una etiqueta con el nombre de la región representada y consta de tres valores: X e Y, que indican la posición de la referencia en la imagen, y Z, un valor aproximado de su distancia a la cámara.

Con los datos obtenidos, a través de las ubicaciones de los puntos de referencia faciales, se cuenta con información suficiente sobre la interacción humano-computador [13]. El arreglo de ubicaciones básicamente simboliza la totalidad del rostro y sus componentes como son la forma, los ojos, las cejas etc.

Es de destacar que cada región es representada por más de un punto. Por ejemplo, del ojo derecho (*right eye*) obtenemos información de la detección, dividida en parte superior y la parte inferior (*right eye upper* y *right eye lower*) y a su vez de cada región recibimos tres puntos de interés (*rightEyeUpper0, rightEyeUpper1* y *rightEyeUpper2* y sus homónimos para la parte baja). La precisión en los datos de cada área permite un cálculo de movimiento más exacto. Estos puntos de referencia son útiles para detectar con mayor precisión la orientación de la mirada del usuario.

En el modelo generado para este trabajo existen básicamente dos grandes pasos: configuración y uso. Para completar la configuración primeramente debemos ingresar la cantidad de monitores con los que iremos a trabajar. Posteriormente, basándonos en el proceso de calibración propuesto por [14] debemos dirigir la mirada a cada una de las pantallas que hemos ingresado en el paso anterior. Una vez terminado, el modelo ya queda listo para detectar hacia cuál de las pantallas estamos dirigiendo la mirada.

3 Prueba práctica en prototipo

El prototipo desarrollado para este estudio trabaja con dos monitores. Para simplificar la configuración se creó una web con dos botones que representan a cada monitor. Fácilmente el usuario puede reconfigurar en cualquier momento, simplemente mirando al monitor que desea, presionar el botón correspondiente y el área en que está mirando queda configurada para la detección. El esquema de funcionamiento es visualizado en la Fig. 2, el usuario ubicado frente a dos monitores, uno de una computadora portátil y otro externo, es registrado por la cámara del ordenador generando así el *streaming* que permitirá la detección del rostro.

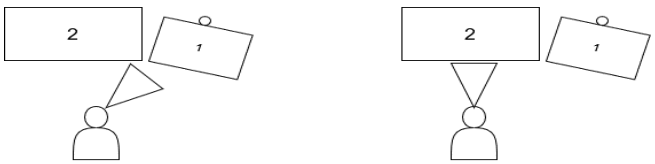

Fig. 2. Esquema de funcionamiento

El proyecto fue desarrollado como una web estática con HTML y *Javascritp* que utiliza el *framework* P5.js para proporcionar la interacción con las imágenes, esta

herramienta facilita, también asignar variables [15] con las que se crea la maya (Fig. 3) sobre el rostro con los principales puntos de referencia y que se va adaptando al mismo mientras se registran sus movimientos en pantalla. Además, se emplea la herramienta *open source* ML5, la cual permite utilizar *TensorFlowJs* de forma más amigable.

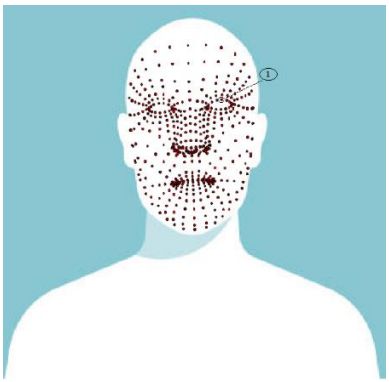

Fig. 3. Esquema de la maya que el modelo cognitivo genera del rostro. El punto 1 destacado en la imagen es la referencia tomada para detectar el movimiento de la cabeza.

El proceso se compone de cinco fases generales. Primeramente, detecta el rostro del operador a través de un *streaming*. Posteriormente, procesa la imagen utilizando el modelo cognitivo de TensorFlow, quien retorna un arreglo con las posiciones del contorno y las distintas partes que conforman el rostro. En este caso, se elige sitio de interés al punto central superior del ojo derecho. Y, finalmente, utilizando los valores ingresados en el proceso de configuración se determina en cuál de las pantallas el usuario tiene el foco, basándose en las coordenadas ingresadas en el paso previo.

Por otro lado, es importante destacar que si el sujeto está portando algún dispositivo o mascarilla la calidad de detección puede disminuir sensiblemente, ya que ciertas partes del rostro no serán localizables haciendo que el seguimiento no sea posible [16]. Esta cuestión es de suma importancia dado que una parte arbitraria de la cara puede ser ocluida por objetos, lo que hace que el algoritmo de detección de puntos de referencia faciales seleccionado deba ser lo suficientemente flexible para manejar diferentes casos (por ejemplo, la boca o la nariz pueden estar ocultas con una máscara) [13].

Tomando en cuenta esto, llegamos a la conclusión de que utilizar una parte del ojo para el cálculo del movimiento puede resultar muy ventajoso. Esto se debe a que la detección realizada por el modelo no se ve afectada si el usuario utiliza anteojos, y además es una parte que estará siempre visible a la cámara. En caso contrario, si hubiéramos optado por trabajar con otras áreas, como la boca o la nariz, la detección podría verse afectada, especialmente si el usuario lleva un barbijo. En esta situación, la información facial disponible se vería reducida, lo que afectaría la capacidad del sistema del modelo para discernir con precisión [17].

Al igual que en este trabajo, el estudio [18] también se centra en la detección del movimiento utilizando puntos de referencia. Sin embargo, a diferencia de este enfoque, [18] emplea seis referencias para llevar a cabo un proceso de calibrado basado en el movimiento de apertura y cierre de ojos y boca. En nuestro caso con tan solo un área del rostro que siempre este visible a la cámara es suficiente. De modo que para

determinar la orientación de la mirada se tomó una sola coordenada (RightEyeUpper0), como punto base para establecer el cambio de pantalla. En lugar de calcular el ángulo de rotación según lo propuesto en [19], en nuestro método solo se considera que el punto de referencia se posicione en cierta área para que se detecte como un salto. Para lograr esto, al presionar cada uno de los botones de configuración se toman los valores X e Y del punto de referencia, quedando esta coordenada como valor que determinará que se está dirigiendo la mirada a esa pantalla.

Para dar cierto margen de movilidad fue conveniente no utilizar el valor puro ya que esto implicaría que el usuario debe pasar su mirada exactamente por el sitio sino no se disparará el cambio de pantalla. Luego de varias pruebas, se decidió establecer una tolerancia de ±20 píxeles. La asignación de esta medida produce alrededor de la coordenada configurada un área de detección más amplia, lo que proporciona una cierta flexibilidad de trabajo.

Es relevante considerar la distancia entre estos dos puntos. Afortunadamente, no enfrentamos el mismo problema reportado en [14], donde el modelo se vuelve ineficiente al intentar mover el cursor entre dos pantallas con distancias muy pequeñas. Para asegurar una detección adecuada, es crucial que los cambios de coordenadas sean detectables en el *streaming*, lo cual está fuertemente influenciado por la calidad de imagen utilizada en el proceso. Además, debemos evitar la superposición de áreas configuradas, ya que esto podría obstaculizar la detección de cambios de región. Una forma de controlar esto es mediante la validación del ingreso de las áreas en tiempo de configuración.

Al igual que se estableció en [14] el uso de este tipo de modelo requiere un tiempo de adaptación por parte del usuario ya que se está sumando una forma más de interacción. Al no tener practica en el uso, puede llegar a acontecer que el operador dirija su mirada al otro monitor y no rote su cabeza, caso que es comentado en [14]. En situaciones como estas el cambio de pantalla no se detectará. Por lo tanto, es recomendable que al momento de configurar las áreas se tenga en cuenta que estas se sitúen lo suficientemente alejadas.

4 Trabajos Relacionados

Se han llevado a cabo numerosos estudios que analizan el proceso de transición entre pantallas, explorando diversas formas de realizar esta tarea. En [2], se detallan los diferentes métodos disponibles para abordar este desafío. No obstante, desarrollar un sistema que facilite esta transición puede ser complejo. Por ello, se han recopilado soluciones que se ocupan de este problema o situaciones similares, siendo la mayoría de ellas enfocadas en mejorar el control del cursor.

Según lo establecidos en [19] dentro de la literatura de investigación existen tres categorías de desarrollo de soluciones para la asistencia HCI. Las que desarrollan un hardware especifico y su software, las que solo desarrollan hardware y las que solo desarrollan software. Con base en las categorías mencionadas anteriormente, se procedió a realizar un análisis de los estudios más recientes.

En el estudio [20], se emplea un giroscopio y sensores para detectar los movimientos de la cabeza, lo que facilita el uso del cursor para personas con discapacidad. Por otro

lado, en [21], se propone un sistema de control remoto universal de bajo costo y código abierto que traduce las poses de la cabeza del usuario en comandos utilizados para controlar dispositivos.

El uso de sensores y hardware especialmente desarrollado también es parte de la solución de [22]. Un sistema de control personas que tienen dificultades para utilizar un mouse convencional. El sistema utiliza un sensor de inclinación en la cabeza para controlar el movimiento del cursor y un botón de soplado que se utiliza para hacer clic. En [23] se presenta un prototipo de sistema para controlar una computadora mediante movimientos de la cabeza y comandos de voz. El sistema utiliza un dispositivo montado en la cabeza que contiene un acelerómetro y un giroscopio para detectar los movimientos que realiza el usuario, además de un micrófono con el que se captan los comandos de voz. Finalmente, toda esta información es enviada a un software que procesa los datos y los traduce ordenes que se ejecutan en la computadora.

A su vez, en [14] se presenta un sistema no intrusivo que utiliza datos de seguimiento de la cabeza para mover el puntero del mouse entre los monitores. El sistema detecta el rostro del usuario con múltiples cámaras y consta de dos etapas: una de inicialización automática de detección del modelo 3D de la cabeza del usuario y otra de seguimiento de esta. En la misma línea se encuentra [24] que describe una interfaz de acceso a la computadora sin manos basada en visión que utiliza una cámara para detectar los movimientos de la cabeza y los gestos de los ojos de los usuarios y traducirlos en acciones del mouse y teclado.

Tal como venimos analizando, existen varias técnicas para detectar los movimientos del rostro, como la propuesta por [25] en la que se desarrolla un hardware especial que se adhiere a los anteojos del operador y se conecta inalámbricamente al ordenador. El dispositivo captura los movimientos a través de sensores y envía los datos al ordenador que los procesa. O en el trabajo de [14] donde no se utilizan sensores, sino que utilizan dos cámaras para tomar en video al operador y con un software desarrollado especialmente, se detecta la orientación del rostro.

Los estudios revisados presentan soluciones innovadoras y accesibles que ofrecen herramientas adicionales para un control más eficiente y efectivo de computadoras y dispositivos. Estos sistemas tienen el potencial de mejorar la interacción. Basándonos en las investigaciones mencionadas, hemos optado por utilizar el paradigma de visión por computadora en nuestra propuesta. Así, nos enfocamos en emplear hardware de uso corriente, como una computadora portátil con una cámara web, para reducir los requisitos técnicos y económicos, y además intentando no ser intrusivos con el operador

En cuanto a la detección y reconocimiento facial, hemos utilizado un modelo que ha demostrado éxito en otros proyectos. Sin embargo, este componente central podría ser reemplazado en el futuro por uno más avanzado, si surgieran modelos cognitivos más eficientes.

5 Conclusiones

En este estudio, se analizaron los elementos esenciales a considerar al crear un modelo para detectar el movimiento facial y determinar a qué pantalla está mirando el usuario en un entorno con múltiples monitores. Se describieron las herramientas utilizadas para

construir un prototipo que facilitó el análisis y la visualización efectiva de los resultados.

La base fundamental de este trabajo se centra en la herramienta de inteligencia artificial *FaceMesh*, la cual permite la detección del rostro y devuelve un arreglo con todos los elementos que lo componen. Cada área del rostro contiene puntos de referencia de los cuales tomamos los tres valores (X, Y, Z) que representan su ubicación en la pantalla. Para el cálculo del movimiento, se requiere utilizar un solo punto de referencia, que debe ser siempre visible para la cámara. Por ello, se decidió trabajar con el área central del rostro, utilizando las coordenadas del ojo derecho como punto de referencia para calcular el movimiento de rotación de la cabeza, tras realizar pruebas para validar esta elección.

El principal inconveniente es que, una vez configurado el sistema, si el usuario se desplaza considerablemente desde el sitio inicial, sus coordenadas X, Y, Z varían notablemente, lo que afecta la precisión de la detección y conduce a resultados incorrectos.

En el prototipo, se propuso una solución mediante la adición de dos comandos para facilitar la reconfiguración cuando el usuario percibe que la detección ya no es precisa. Para futuros trabajos, se busca desarrollar un proceso de configuración automático que recalcule las áreas de detección de los monitores basándose en las coordenadas iniciales y la nueva posición del operador. Además del movimiento en los ejes X e Y, también puede darse el caso de que el usuario se aleje considerablemente de la cámara, afectando la precisión en el eje Z. En esta situación, la solución de reconfiguración automática sería útil, aunque se debe tener en cuenta que el valor de Z es estimado debido a que el sistema cuenta con una sola cámara, lo que podría generar cierta imprecisión.

No obstante, es importante tener en cuenta que el valor de Z es estimado debido a que, al utilizar una sola cámara, la paralaje no es totalmente precisa. A pesar de esto, es posible analizar cómo recalcular la ubicación, teniendo en cuenta la limitación de precisión mencionada anteriormente.

6 Referencias

1. Robertson, G., Czerwinski, M., Baudisch, P., Meyers, B., Robbins, D., Smith, G., Desney Tan: The Large-Display User Experience. IEEE Comput Graph Appl. 25, 44–51 (2005).
2. Desrosiers, T., Livingston, I., Noete, D., Wourms, N.: Cross Display Mouse Movement in MDEs.
3. Czerwinski, M., Robertson, G., Meyers, B., Smith, G., Robbins, D., Tan, D.: Large display research overview. In: CHI ’06 Extended Abstracts on Human Factors in Computing Systems. pp. 69–74. ACM, New York, NY, USA (2006).
4. Saleem, J.J., Weiler, D.T.: Performance, workload, and usability in a multiscreen, multi-device, information-rich environment. PeerJ Comput Sci. 4, e162 (2018).
5. Baudisch, P., Cutrell, E., Robertson, G.: High-Density Cursor: A Visualization Technique That Helps Users Keep Track of Fast-Moving Mouse Cursors. In Interact. Vol.3, 236–243 (2003).
6. Nacenta, M.A., Mandryk, R.L., Gutwin, C.: Targeting across displayless space. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 777–786. ACM, New York, NY, USA (2008).

7. Murphy-Chutorian, E., Trivedi, M.M.: Head Pose Estimation in Computer Vision: A Survey. IEEE Trans Pattern Anal Mach Intell. 31, 607–626 (2009).
8. Kuhnke, F., Ostermann, J.: Domain Adaptation for Head Pose Estimation Using Relative Pose Consistency. IEEE Trans Biom Behav Identity Sci. 1–1 (2023).
9. Li, D., Cui, Z., Cao, F., Cui, G., Shen, J., Zhang, Y.: Learning State Assessment in Online Education Based on Multiple Facial Features Detection. Comput Intell Neurosci. 2022, 1–16 (2022).
10. Chuan, T., Xinrui, H., Zhicheng, W., Yu, Z., Mingyu, X., Xin, W.: Head Pose Estimation via Multi-Task Cascade CNN. In: Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference. pp. 123–127. ACM, New York, NY, USA (2019).
11. Abate, A.F., Bisogni, C., Castiglione, A., Nappi, M.: Head pose estimation: An extensive survey on recent techniques and applications. Pattern Recognit. 127, 108591 (2022).
12. Shao, X., Qiang, Z., Lin, H., Dong, Y., Wang, X.: A survey of head pose estimation methods. In: 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). pp. 787–796. IEEE (2020).
13. Wu, Y., Ji, Q.: Facial Landmark Detection: A Literature Survey. Int J Comput Vis. 127, 115–142 (2019).
14. Ashdown, M., Oka, K., Sato, Y.: Combining head tracking and mouse input for a GUI on multiple monitors. In: CHI '05 Extended Abstracts on Human Factors in Computing Systems. pp. 1188–1191. ACM, New York, NY, USA (2005).
15. Harditya, A.: Indonesian Sign Language (BISINDO) As Means to Visualize Basic Graphic Shapes Using Teachable Machine. In: In International Conference of Innovation in Media and Visual Design. pp. 1–7 (2020).
16. Abdullah, D.B., Alnuaimy, M.: Real-time Face Tracking for Service-Robot. Technium: Romanian Journal of Applied Sciences and Technology. 4, 47–52 (2022).
17. Jeevan, G., Zacharias, G.C., Nair, M.S., Rajan, J.: An empirical study of the impact of masks on face recognition. Pattern Recognit. 122, 108308 (2022).
18. Ming, C., Yunbing, Y.: Perception-Free Calibration of Eye Opening and Closing Threshold for Driver Fatigue Monitoring. IEEE Access. 10, 125469–125476 (2022).
19. Nelson Garrido, Neil, C., Battaglia, N.: Tecnología asistiva para personas con discapacidad en miembros superiores: un mapeo sistemático de la literatura. (2022).
20. Kim, S., Park, M., Anumas, S.: Head Mouse System Based on Gyro- and Opto- Sensors. 1503–1506 (2010).
21. Batista, C.T., Campos, E.M., Neto, N.C.S.: A proposal of a universal remote control system based on head movements. In: ACM International Conference Proceeding Series. Association for Computing Machinery (2017).
22. Mishra, M., Bhalla, A., Kharad, S., Yadav, D.: HMOS : Head Control Mouse Person with Disability. 576–583 (2017).
23. Ismail, A., HAJJAR, A.E.S. AL, HAJJAR, M.: A prototype system for controlling a computer by head movements and voice commands. arXiv preprint arXiv:1109.1454. 3, 15–25 (2011).
24. Varona, J., Manresa-Yee, C., Perales, F.J.: Hands-free vision-based interface for computer accessibility. Journal of Network and Computer Applications. 31, 357–374 (2008).
25. Bender Machado, M., Sias Rodrigues, A., Bender Machado, M., Kruger Da Costa, V., Cunha Cardoso, R., Souza Medeiros Quadros, C.L., Ferreira Xavier, K., Peroba, J., Aires Tavares, T.: An adaptive hardware and software based human computer interface for people with motor disabilities. IEEE Latin America Transactions. 17, 1401–1409 (2019).