Interacció amb moviment del cap: Estudi qualitatiu i quantitatiu de l'estat de l'art

Ramon Aranda Ramírez

Resum– La interacció humà-computador és una de les principals àrees d'investigació de la Visió per Computació. En aquest projecte es presenta un estudi quantitatiu i qualitatiu dels diferents mètodes de l'estat de l'art de Head/Face Tracking en el context d'interacció amb el cap utilitzant únicament la llum natural. Primerament, s'estudia la bibliografia disponible i s'analitza el rendiment dels softwares lliures més rellevants. Aquests softwares són posats a prova utilitzant mètriques de forma qualitativa i quantitativa per tal d'escollir el candidat òptim per a utilitzar-ho en un entorn hardware limitat com una Raspberry Pi amb una càmera estàndard. Finalment, s'implementa un prototipus de software funcional com a aplicació de validació del mètode.

Paraules clau– Face Tracking, Head Tracking, Estudi quantitatiu, Estudi qualitatiu, Interacció humà-computador.

Abstract– Human-Computer interaction is one of the main areas of research of Computer Vision. In this project we present a quantative and qualitative study of different state-of-the-art methods for Head/Face Tracking in the context of head interaction using natural light only. First, available bibliography is studied and the performance of the most relevant open source softwares are analized. Those softwares are tested using both qualitative and quantitative metrics in order to select the optimal candidate to be used in a hardware-limited environment such as a Raspberry Pi with a standard webcam. Finally, a functional software prototype is implemented as a validation aplication for the method.

Keywords– Face Tracking, Head Tracking, Quantitative study, Qualitative study, Human-Computer interaction.

1 Introducció

Imaginar-se un món en el qual es pogués interactuar amb els dispositius sense la necessitat de perifèrics a part d'una càmera, és un dels principals problemes, que des de fa molts anys, la visió per computador intenta resoldre de diferents formes.

Existeixen solucions molt diverses com per exemple Face Tracking, Head Tracking o Eye Tracking. En aquest context, el principal objectiu de l'estudi d'aquest projecte és la captació de l'atenció visual a partir de l'anàlisi de la posició de la cara. Donat el gran abast de recerca d'aquest tema, aquest projecte acotarà el seu estudi quantitatiu i qualitatiu en les solucions de Head Tracking i Face Tracking amb llum natural.

A més, les diferents aplicacions de la solució d'aquest problema envers el món real, recullen un gran nombre de facilitats per a les persones a l'hora interactuar amb el seu entorn, des d'ajudar a persones discapacitades a utilitzar dispositius, fins a facilitar la feina de la vida quotidiana d'una manera regular.

Aquest article està estructurat en 5 seccions diferents: La primera secció recull una petita introducció de l'article juntament amb la motivació personal del projecte i els diferents objectius d'aquest. En la segona secció es troba tota la informació pertinent a l'estudi de l'estat de l'art actual sobre els diferents softwares de Head/Face Tracking. En la tercera secció s'explica quines mètriques s'utilitzaran per a avaluar els diferents softwares de l'estat de l'art, els diferents *set-ups* dels experiments i la fase d'experimentació del projecte. La quarta secció recull com s'extreu els resultats de tota la informació obtinguda dels experiments i com s'avaluen aquests resultats. Per últim, en la cinquena secció es pot observar les diferents conclusions extretes dels resultats

E-mail de contacte: ramon.aranda.r@gmail.com
Menció realitzada: Computació
Treball tutoritzat per: Fernando Vilariño (Computació)
Curs: 2014/15

obtinguts en la secció anterior, com es podria millorar i ampliar el projecte en un futur i els agraïments pertinents a les persones que han fet possible el desenvolupament d'aquest projecte.

1.1 Motivació

La principal motivació del projecte és la curiositat envers l'estudi i enteniment del funcionament dels diferents softwares de Head/Face Tracking de l'actualitat. Un altre motiu és que actualment no hi han gaires comparatives de softwares de l'estat de l'art respecte al tema dels Head/Face Tracking. Parlant des d'un punt de vista personal, la idea de poder aportar al món una comparativa dels diferents softwares actuals va ser de grat. D'aquesta forma, l'aportació d'aquest projecte al món és, encara que per petita que sigui, una guia o pauta per aquelles persones que busquin començar a fer un projecte o vulguin utilitzar un software de l'estat de l'art per a fer una aplicació funcional i/o millorar-lo. La segona motivació personal ve en intentar crear un prototipus funcional, sigui un joc o una aplicació, com a mecanisme de validació experimental per a donar a conèixer l'aprenentatge dels diferents mètodes i la contribució personal al software.

1.2 Objectius

L'objectiu principal d'aquest treball és estudiar quins són els softwares de Head/Face Tracking amb llum natural, sense l'ajut de cap dispositiu extern, que més robustesa i rendiment tenen en l'actualitat. Aquest objectiu generalitzat es pot dividir en diferents subobjectius. Per tal de poder donar per assolit l'objectiu principal, els següents subobjectius també han d'estar assolits completament:

- Estudiar l'estat de l'art actual de Head/Face Tracking.
- Fer una recerca quantitativa dels diferents softwares més comuns o utilitzats que no apliquin l'ajut de cap dispositiu extern, és a dir, només llum natural.
- Estudiar quines mètriques s'han d'utilitzar per a avaluar els softwares trobats en el subobjectiu anterior.
- Fer l'estudi qualitatiu aplicant aquestes mètriques als softwares per a obtenir-ne resultats.
- Avaluar els diferents resultats per tal de trobar quin és candidat que dóna millor rendiment amb les mètriques aplicades.
- Avaluar la viabilitat d'utilitzar el software candidat en un dispositiu Raspberry Pi. En altres paraules, l'objectiu és trobar un estàndard de requeriments mínims en limitacions hardware per a poder escalar el software.

El segon objectiu del projecte és crear un prototipus funcional per tal de validar el mètode escollit en el primer objectiu. Així, la finalitat de l'aplicació és una aplicació de validació experimental de la interacció guiada de l'atenció de l'usuari a partir de la posició del cap/cara. Un bon exemple, i en el que es basa aquest projecte, és el joc de "On està Wally?".

2 Bibliografia de Head Tracking i Face Tracking

2.1 Principals referències de l'estat de l'art

En l'àmbit de recerca de la interacció amb el moviment del cap basat en Head/Face Tracking existeixen diverses implementacions. Hi ha d'algunes que es basen en algun dispositiu extern, altres en LEDs infrarojos i d'altres que utilitzen la llum natural. Ara bé, gairebé totes les implementacions tenen certes limitacions en comú. Per exemple, la implementació restringeix les poses que l'usuari pot fer perquè el software no perdi el model, és a dir, per tal de poder capturar les característiques de l'usuari, aquest ha d'estar mirant la càmera o amb certes poses limitades. En general, totes són molt sensibles als canvis de la llum ambient [27] i de les característiques de la fisonomia física dels usuaris.

Pel que fa a les implementacions que utilitzen dispositius externs, encara que tots es basin a utilitzar LEDs infrarojos, utilitzen un tipus de hardware específic. Un bon exemple d'això són les implementacions que es basen en Wiimote [28, 29]. A través del hardware que es disposa de la consola Wii, s'utilitza el seu comandament per a localitzar en l'espai a l'usuari gràcies a detector d'infrarojos que proveeix la mateixa consola. D'altra banda, existeixen implementacions comercials com TrackIR [15] que utilitzen una càmera i un conjunt de LEDs infrarojos per a detectar la posició i orientació de l'usuari en l'espai. No obstant això, hi ha versions lliures com Free-Track [14] que només amb 1 LED donen un model amb 3 graus de llibertat (posició en l'espai) i amb 3 LEDS un model de 6 graus de llibertat (posició i orientació).

En relació amb els que funcionen amb una càmera i llum natural, hi ha moltes variacions en termes d'implementacions, però en general tots segueixen un mateix patró. Primer, calculen la posició de l'usuari utilitzant algun sistema de detecció de features o cares, com ara bé Viola Jones [31, 32] o Moghaddam and Pentland [19]. Tot seguit, calculen un model 2D o 3D del reconeixement, on la implementació d'aquesta part és molt variada, i encaixar-lo al cap/cara de l'usuari. Seguit, es pot implementar millores del model utilitzant algun algoritme de Landmark Estimation, com AAM [20] o Constrianed Local Models [22], entre d'altres. Aquests algoritmes es basen a trobar els punts de referència de la fisonomia de la cara de la persona, per a poder calcular el model i aplicar-ho a la imatge amb més precisió.

2.2 Conjunt de software triat per a l'estudi

Dels principals mètodes de l'estat de l'art, es trien implementacions de Head/Face Tracking lliures que funcionen amb llum natural, sense l'ajut de cap dispositiu extern excepte una webcam.

Ehci [10, 11] i cvHead [16] són implementacions de Head Tracking amb Head Pose Estimation. Utilitzen el detector Viola Jones per a detectar la cara de l'usuari i la seva posició. Seguit, creen un model i l'apliquen a la imatge 2D capturada de la càmera. Després, utilitzen Lucas-Kanade per a seguir els punts dels fotogrames per a actualitzar la matriu de posició de l'usuari. A diferència de cvHead, ehci aplica RANSAC [35, 36] per tal de millorar les mostres del

seu model.

fTracker [1, 2, 3, 4] és una implementació de Face Tracking que utilitza Viola Jones per detectar la cara. Seguit, crea el model utilitzat un Point Distribution Model (Landmark Estimation) i el situa a la imatge capturada del dispositiu.

PyHead i PyFace [12, 13] són implementacions de Head Tracking i Face Tracking respectivament que utilitzen el detector de Viola Jones per a crear el seu model i aplicar-ho a la imatge capturada.

3 METODOLOGIA

3.1 Desenvolupament del software

La metodologia aplicada al desenvolupament del projecte és l'àgil [23]. Cada iteració de la metodologia consisteix en una fase de planificació, anàlisi dels requisits, disseny, codificació, revisió i documentació. Al final de cada iteració, es mostra una demostració dels avenços fets. D'aquesta manera, el software que es va desenvolupant durant les iteracions tendeix a desviar-se menys del seu propòsit inicial i és molt flexible al canvi d'expectatives o modificacions noves. Així, cada objectiu, o subobjectiu en cas que l'objectiu en tingui, és una iteració d'aquesta metodologia.

3.2 Infraestructura del software

Abans de muntar tots els experiments per a avaluar els softwares, cal establir entorns controlats amb una limitació hardware determinada i amb una fàcil recuperació en cas que deixessin de funcionar degudament. Per tal de fer això, la solució òptima és crear un conjunt de màquines virtuals amb les instal·lacions de frameworks pertinents. Com que aquests estan programats per diferents plataformes (Linux i Windows), es crea un entorn virtual d'Ubuntu 14.04 per als softwares cvHead i fTracker, i un entorn de Windows 7 per als softwares ehci, pyHead i pyFace amb les mateixes limitacions hardware.

Amb el fi de dur a terme els experiments, es dissenya i s'implementa un mòdul software que permet automatitzar les proves de les mètriques i guardar-ne el resultat. El mòdul consisteix en un conjunt de classes amb la finalitat d'interaccionar amb el software per a extreure'n resultats i guardar-lo en un sistema de logs.

Aquest mòdul no és únic per a tots els softwares, ja que a banda del fet que estan implementats en diferents plataformes, també estan programats en diferents llenguatges de programació. Doncs, cal replicar el funcionament d'aquest mòdul per totes les plataformes i/o llenguatges de programació, és a dir, implementar-ho de nou per a aquestes plataformes i llenguatges de programació. Ehci treballa en Windows i C++, pyHead i pyFace en Windows i Python i, fTracker i cvHead en Linux i C++. Per tant, cal implementar 3 mòduls independents per a poder avaluar tots els softwares proposats.

3.3 Mètriques

Com que el principal objectiu és trobar quin és el software més robust i quin té un menor percentatge de consum dels diferents recursos d'un ordinador, l'estudi de les mètriques a aplicar es dividirà en dos grups principals: les mètriques de rendiment i les mètriques de robustesa.

Mètriques		
Tipus	**Número**	**Descripció**
Rendiment	M1	Rapideça de l'algoritme (milisegons).
	M2	Fotogrames per segon (FPS).
	M3	Resolució dels frames.
	M4	Percentatge de CPU.
	M5	Quantitat de RAM (MegaBytes).
	M6	Quantitat de Virtual RAM (MegaBytes).
Robustesa	M7	Oclusions.
	M8	Angle màxim per detectar la cara o cap.
	M9	Angle mínim per detectar la cara o cap.
	M10	Percentatge d'error en la rotació pitch.
	M11	Percentatge d'error en la rotació yaw.

Taula 1: Conjunt de mètriques a aplicar als softwares.

Les mètriques de rendiment són aquelles que mesuren la quantitat de recursos de l'ordinador que està consumint el software en plena execució i la rapidesa d'aquest. El conjunt de factors que el software consumeix del hardware són: el percentatge de CPU, la quantitat de RAM i la quantitat de Virtual Ram. Així mateix, el rendiment per part del software, deixant de banda el consum de hardware, es pot mesurar en la quantitat de frames per segon que és capaç d'utilitzar, el temps que triga a executar-se una iteració del programa i la resolució dels frames.

D'altra banda, les mètriques de robustesa són aquelles que mesuren les limitacions dels softwares. En altres paraules, la principal intenció d'aquestes mètriques és portar els diferents softwares a avaluar fins als seus límits, per a conèixer quins són aquests per tal de saber fins a quin punt la interacció de la mirada a partir de la posició del cap/cara és factible. Així, cal estudiar els límit de les rotacions en els eixos X i Y, obtenint quin és l'error del model en les diferents rotacions i quin és el seu límit. Tanmateix, hi ha un factor molt important que també s'ha de tenir en compte, que són les oclusions.

El conjunt de mètriques a aplicar per a avaluar els diferents softwares queden recollides a la Taula 1.

3.3.1 Mètriques de rendiment

Primerament es duu a terme l'avaluació de M1. Aquesta, consisteix a calcular el temps que triga el software a fer una iteració completa. Per a iteració es sobreentén que és des del moment que el programa comença a fer el reconeixement de cara/cap, fins que finalitza tot el procés general. Durant un mateix segon, pot haver-hi més d'una iteració. Per tal d'aconseguir calcular el temps que triga cadascuna d'aquestes iteracions a dur-se a terme, cal crear un temporitzador. Aquest temporitzador consisteix en dues variables: la primera recull el temps inicial en el qual s'engega el temporitzador (abans de començar la iteració) i la segona el temps final quan es para aquest temporitzador. Amb els dos valors, cal aplicar una diferència absoluta per a veure el temps que ha trigat a executar-se aquesta iteració.

L'avaluació de M2 rau a calcular el nombre de fotogrames per segon (FPS) a què cada software pot funcionar. La quantitat de FPS ve donada per la quantitat d'iteracions que el software pot fer en un segon. Doncs, per calcular-ho s'u-

tilitza l'Equació 1:

$$FPS = \frac{1}{IterationTime} \quad (1)$$

M3 és un indicador del límit del software envers la resolució. Aquesta part per si sola no aporta gaire valor als resultats, però juntament amb la primera part, es pot crear una petita hipòtesi per a avaluar els sistemes que radica en si la velocitat del software és independent de la resolució dels frames. Doncs, per a obtenir resultats en aquest apartat, cal repetir la primera part tants cops com resolucions es fiquen a prova al software, una per cada resolució diferent.

Finalment es mesura la quantitat de recursos que el software està utilitzant del sistema, és a dir, M4, M5 i M6. Per tal d'obtenir-ho, a cada mòdul s'utilitza una llibreria diferent, un per a cada cas, per tal d'accedir als valors dels recursos del sistema, obtenir-ne els valors desitjats i guardar-los en el sistema de logs.

3.3.2 Mètriques de robustesa

D'altra banda, pel que mètriques de robustesa es tracta, primerament s'observa si el model permet o no oclusions. Per tal d'avaluar aquesta part, és a dir, M7, l'usuari mira fixament a la webcam mentre que s'apliquen certes oclusions. Les oclusions més comunes en les persones són en la boca, per exemple amb bufandes o mocadors, i en els ulls, per exemple les ulleres. Per tant, les diferents oclusions que es posen a prova són:

1. Ocultar la boca.
2. Ocultar un quart de la cara (fins al primer ull d'aquella banda).
3. Ocultar mitja cara (fins al nas).
4. Ocultar tres quarts de la cara (fins al segon ull).

El principal motiu del perquè aquestes oclusions és perquè la gran majoria de softwares de l'estat de l'art agafen com a punts de referència els ulls, la boca, el nas i la perifèria de la cara [34]. Com que l'objectiu de l'estudi és portar fins al límit els diferents softwares, d'aquesta manera s'estan forçant per a esbrinar fins a quin punt cada model permet certes oclusions sense que el model es vegi afectat, és a dir, es deformi.

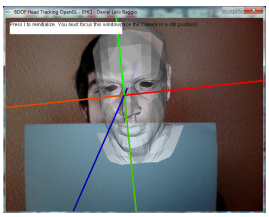
Per acabar, s'avaluen les mètriques restants, en altres paraules, M8, M9, M10 i M11. El motiu del perquè estan totes juntes en un mateix conjunt, és perquè totes rauen en l'avaluació de les rotacions dels models dels softwares. Com que només ehci i cvHead permeten el càlcul del vector d'atenció, l'estudi d'aquesta mètrica s'aplica de forma diferent per als que ho permeten i per als que no.

- Respecte als softwares que calculen el vector d'atenció, l'objectiu és calcular els angles d'aquest vector amb la finalitat de poder comparar-los amb l'angle que l'usuari avaluador fica a prova en aquell moment, és a dir, calcular l'error del càlcul de l'angle del software. Per tal de dur a terme el calcul dels angles, primer s'ha d'aïllar i destransposar la matriu de rotació de la matriu de transformació, que està situada a la part superior esquerra de la matriu de transformació tal com es pot veure a l'Equació 2. Un cop obtinguda la matriu de rotació, a l'Equació 3 es detalla com aconseguir els angles d'Euler a partir de la matriu de rotació [6, 7].

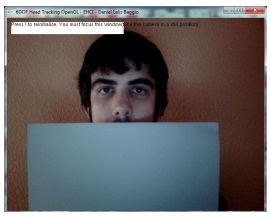
$$M = \begin{bmatrix} R' & T \\ 000 & 1 \end{bmatrix} \quad (2)$$

$$\begin{gathered} R = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \\ \theta_x = atan2(r_{3,2}, r_{3,3}) \\ \theta_y = atan2(-r_{3,1}, \sqrt{r_{3,2}^2 + r_{3,3}^2}) \end{gathered} \quad (3)$$

- Pel que fa els softwares que no ho permeten, per tal d'avaluar-los, com no es té accés a la matriu de transformació o rotació del model, s'anota manualment amb valor verdader si el model no s'ha deformat en aquell angle i fals si ho ha fet.



(a)



(b)

Figura 1: Exemple d'oclusió. A la figura (a) es mostra l'oclusió amb el model superposat, mentre que a la (b) només es mostra l'oclusió.

3.4 Fase d'experimentació

Abans de dur a terme l'experiment de M1, M2 i M3, cal definir quines són les resolucions que s'utilitzaran i avaluaran. Per dur a terme tots els experiments s'utilitza una webcam Logitech C270 que té una resolució màxima de 1280x720, per tant no es pot superar aquest valor a l'hora d'escollir les resolucions. Partint d'aquesta base, les resolucions més comunes amb un ràtio de 4:3 són: 640x480 (VGA; per defecte a OpenCV), 800x600 (SVGA) i 1024x768 (XGA). Les resolucions amb un ratio de 16:9 són: 854x480 (WVGA) i 1280x720 (HD 720).

Un cop definides les resolucions a aplicar, per dur a terme l'experimentació M1, M2 i M3, s'executa cada software 5 cops, cada cop amb una resolució diferent. Per tal de no obtenir un volum massa gran de dades ni un de massa petit, la durada de cada execució és de 2 minuts i 30 segons. Aquest valor s'ha triat mitjançant proves prèvies trivials.

Per tal d'avaluar M4, M5 i M6, s'executa cada un dels softwares durant 2 minuts i 30 segons amb una resolució de 640x480 (resolució per defecte) i es deixa que el mòdul s'encarregui de fer el log de les dades dels diferents recursos del sistema.

Per posar a prova M7, s'executa el software mentre que a l'usuari avaluador se li apliquen certes oclusions, tal com es pot veure a la Figura 1. Si durant el transcurs de l'oclusió el model es veu deformat, s'anota un negatiu en aquell test, mentre que si no es deforma, s'anota un positiu.

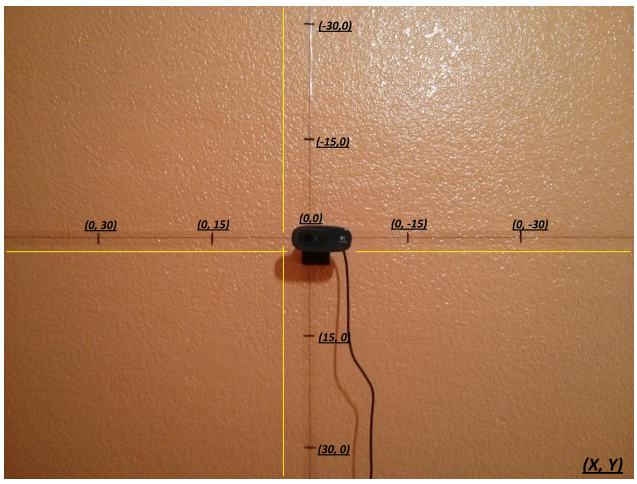


Figura 2: Escenari per a posar a prova el conjunt d'angles. Es mostren com a exemple les marques de -15, 15, -30 i 30 graus en els eixos X i Y.

Per tal de poder avaluar les últimes mètriques, es crea un escenari on l'usuari avaluador tingui punts de referència fins a on ha de moure el cap, tal com mostra la Figura 2. Per poder crear aquest escenari, primer s'escullen quins angles es posaran a prova, que en aquest cas el conjunt d'angles és: -60, -45, -30, -15, 0, 15, 30, 45 i 60. El límit dels angles es fixa en -60 i 60 graus, ja que per limitacions físiques a la persona li costa moure el cap més d'aquests angles, però també cal tenir en compte que si una persona es posa a una distància prudent d'un monitor, mai superarà aquest angle de gir. La idea central és saber a quin punt del pla de la paret intersecciona el vector d'atenció de l'usuari per a cada angle. Per tal de calcular-ho, s'aplica la llei de sinus [24] tal com es mostra a l'Equació 5 després d'aïllar els valors de l'Equació 4, que és l'original.

$$\frac{a}{sin(\alpha)} = \frac{b}{sin(\beta)} = \frac{c}{sin(\gamma)} \tag{4}$$

$$\beta = \frac{x * sin(y)}{sin(90 - y)} \tag{5}$$

A l'Equació 5, el valor x és la distància entre la webcam i l'usuari avaluador, en aquest cas la variable té un valor constant de 50 cm i el valor y és l'angle que es posa a prova (en valor absolut).

Un cop definit l'escenari per a posar a prova les mètriques restants, es defineix com s'avaluen aquestes, ja que hi ha alguns softwares que implementen Head/Face Pose Estimation i d'altres que no. Per als que ho fan, amb l'ajut del mòdul software implementat, l'usuari pot decidir quan guardar els angles que calcula el software mitjançant una tecla. Doncs, un cop l'usuari avaluador gira el cap fins a l'angle a provar, li fa saber al mòdul que ha de guardar el càlcul dels angles d'aquell moment al sistema de logs. En canvi, per als softwares que no ho gaudeixen, s'anota quin és l'angle on el model es deforma.

4 Resultats

4.1 Processat i visualització dels resultats

Per tal de poder processar correctament els resultats i preparar-los per a l'avaluació, cal preprocessar-los amb l'objectiu que més endavant siguin fàcilment manipulables per a extreure'n conclusions de forma més senzilla.

El primer pas és disminuir el nombre de fitxers de log actuals. Amb aquest propòsit, s'implementa un mòdul, anomenat "parser". Els tres principals objectius d'aquest mòdul són: El primer pas és implementar un mòdul software anomenat *parser* que té com a objectius:

- Llegir el sistema de logs per a obtenir-ne les dades.
- Agregar els resultats utilitzant una mitjana..
- Guardar les dades en un sistema d'emmagatzemament fàcilment accessible i òptim, en altres paraules, una base de dades.

El funcionament d'aquest mòdul queda recollit en la Figura 3.

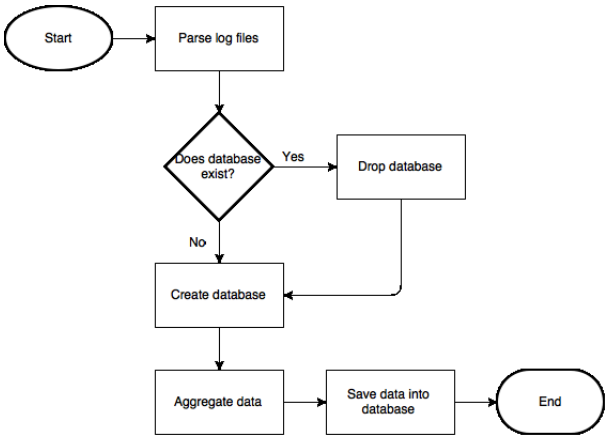


Figura 3: Funcionament del mòdul *parser*.

El segon pas és implementat per un altre mòdul, anomenat *plotter*, amb l'objectiu de crear gràfics per a ajudar a avaluar els resultats dels diferents experiments. Aquests gràfics ajuden a entendre millor, dins la perspectiva qualitativa, els resultats numèrics que només visualitzant grans volums de dades [25, 26]. El funcionament d'aquest mòdul queda recollit a la Figura 4.

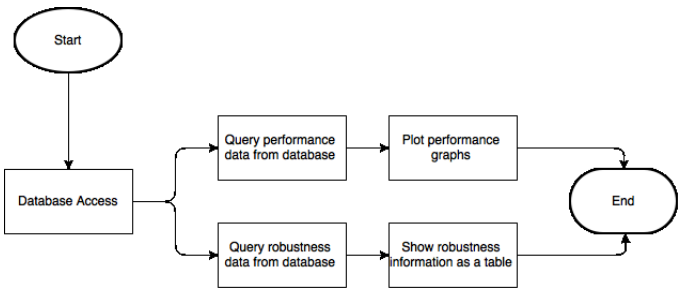


Figura 4: Funcionament del mòdul *plotter*.

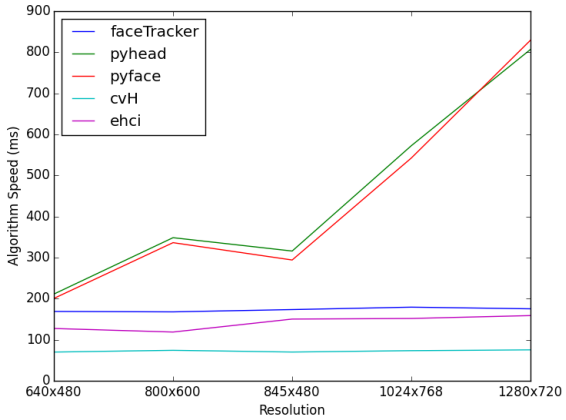


Figura 5: Resultat de la correspondència de M1 i M3. Velocitat del software respecte a la resolució dels frames.

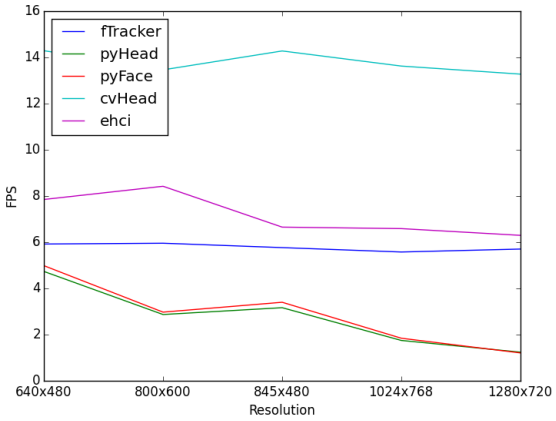


Figura 6: Resultat de la correspondència de M1 i M2. Quantitat de fotogrames per segon respecte la resolució dels frames.

4.2 Analisi els resultats

4.2.1 Resultats de les mètriques de rendiment

Primerament, els resultats de M1, M2 i M3 queden reflectits en la Figura 5 i Figura 6. A la Figura 5 es pot observar clarament com els tres softwares que funcionen sobre C++ no hi ha variació en la durada d'una iteració del software respecte a la resolució, però els dos que treballen en Python sí que ho fan. Hi ha molts motius del perquè, com ara bé que C++ és un llenguatge compilat mentre que Python és un llenguatge interpretat. En resum, Python és un llenguatge que guanya en simplicitat, però per a tenir un bon rendiment és molt millor C++.

Tant pyFace i pyHead utilitzen arrays de Python en la seva implementació. Aquest és un dels pitjors casos en termes de gestió de memòria, ja que les arrays de Python estan construïdes com a objecte i no com a primitives, fent que a l'hora de tractar-les sigui més lent. Per tal d'evitar aquest problema, cal utilitzar la llibreria numpy [33] que és un wrapper de les arrays de C++, que són primitives i estan optimitzades per a treballar més rapidament amb elles.

Com es possible veure a la Figura 6, cvHead és el que més frames per segon aconsegueix, però no perquè tingui més FPS vol dir que sigui millor software, ja que per deduir això cal observar els resultats de la resta de mètriques. El següent punt a considerar són les mètriques restants de rendiment, és a dir, aquelles que mesuren el consum del software respecte al hardware. Cal recordar que aquesta prova es fa amb la resolució per defecte que és 640x480.

Respecte del consum de CPU, a la Figura 7 es plasmen els resultats de la prova. El millor software en aquest cas és ehci, tot seguit pel conjunt de pyHead i pyFace amb un consum molt pròxim al 10% de la CPU. Sobre els softwares fTracker i cvHead, el seu consum és alt, ja que si se suma el consum del software juntament amb el qual consumeixi el sistema operatiu, que per poc que sigui sempre consumeix, se supera amb escreix el límit de la Raspberry Pi. En conseqüència, tant fTracker com cvHead no són candidats òptims per a fer-los servir en tal dispositiu.

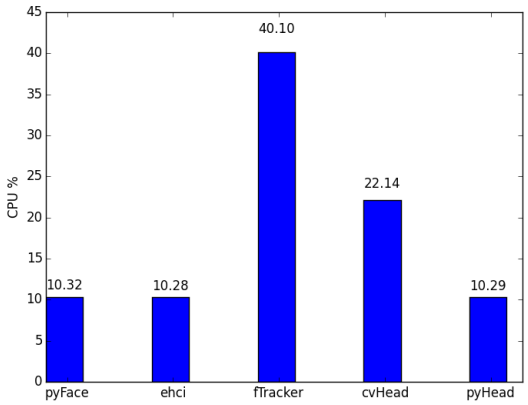


Figura 7: Resultat de M4. Quantitat de CPU consumida pel software (en percentatge).

Un altre punt important és el consum de memòria, tant de RAM com VRAM, on els resultats es mostren a la Figura 8 i la Figura 9 respectivament. Pel que fa el consum de RAM, fTracker és el millor amb diferència, tot seguit de ehci i cvHead. No és cap casualitat que aquests tres softwares estiguin per sobre de pyHead i pyFace, pel que consum de memòria es tracta. Un dels motius és el mateix que s'ha explicat al principi d'aquest apartat, i és que les arrays de C++ són primitives mentre que les de Python són objectes. Aquesta array de Python pot emmagatzemar diferents tipus d'objectes en una mateixa array. Per tant per tal de saber quin tipus de dades hi ha en cada posició, cal guardar de quin tipus de dada es tracta fent així que les arrays ocupin més de per si. Hi pot haver altres motius que, per esbrinarlos, cal indagar en el codi profundament cosa que no es farà, ja que no és un dels objectius principals del projecte.

Sobre el consum de Virtual RAM, es pot veure que hi ha una clara diferència entre els softwares que funcionen en Windows (ehci, pyFace i pyHead) amb els que funcionen en Linux (fTracker i cvHead). No obstant això, ehci torna a ser el millor, seguit de pyFace i pyHead. A partir d'analitzar aquests resultats, sorgeix la hipòtesi de si és possible que hi hagi una diferència entre córrer el software en un sistema operatiu en un altre. Si fos així, s'hauria d'estudiar quin és el millor d'aquest, però com que això s'escapa dels objectius inicials del projecte, no es tindrà en compte en l'avaluació actual, però si quedarà constància com a ampliació

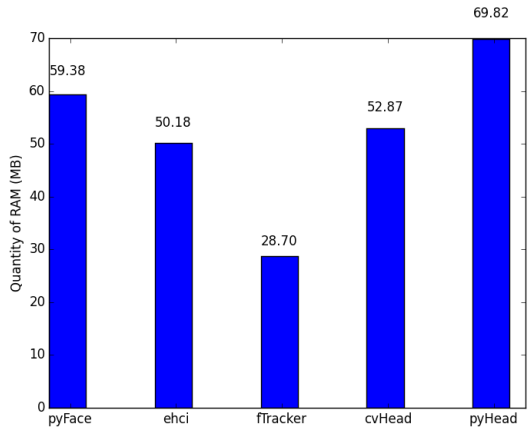


Figura 8: Resutlat de M5. Quantitat de RAM consumida pel software (en MB).

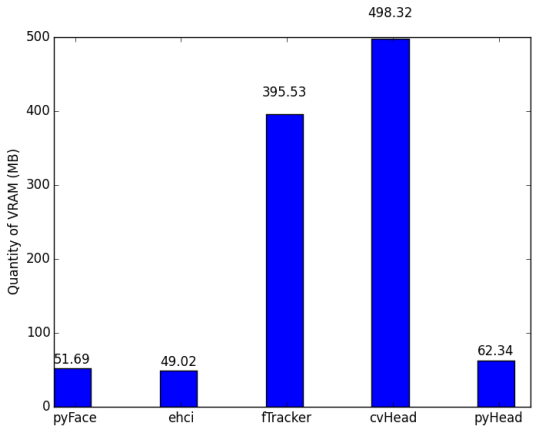


Figura 9: Resultat de M6. Quantitat de VRAM consumida pel software (en MB).

del projecte.

Així doncs, els resultats de l'avaluació dels softwares de les mètriques de rendiment queden recollits en la Taula 2 on els valors són les posicions del software respecte al resultat d'aquella mètrica. El millor software en termes de rendiment i consum és ehci.

4.2.2 Resultats de les mètriques de robustesa

Pel que respecta la M7, el resultat obtingut és que cap dels softwares posats a prova permet oclusions a escala general. No obstant això, cvHead sí que permet oclusions en l'àmbit de la boca. El motiu és que aquests agafa com a punts de referència per al model punts del nas i dels ulls de l'usuari, i cap de la perifèria de la cara o de la boca. D'aquesta forma, si s'oculta la boca, el model segueix funcionant per igual.

	ehci	cvHead	fTracker	pyFace	pyHead
Velocitat	2	1	3	4	5
FPS	2	1	3	4	5
CPU	1	4	5	3	2
RAM	2	3	1	4	5
VRAM	1	5	4	2	3
Total	**8**	**14**	**16**	**17**	**20**

Taula 2: Resultats de les mètriques de rendiment. Rànquing dels algoritmes respecte dels resultats de les mètriques.

Es recullen els resultats de les mètriques de robustesa restants en diferents taules. Pel que fa els softwares de Head/Face Tracking que implementen Head/Face Pose Estimation es considera acceptable un error d'un 20% en les mostres. En canvi, pels softwares que no ho implementen es considera com a vàlid fins al primer error en la mostra dels angles, és a dir, l'angle on es trobi el primer error en la mostra serà l'angle màxim/mínim d'aquell rang.

A la Taula 3 i Taula 4 es troben els resultats dels dos softwares de Head Tracking amb Head Pose Estimation, dit d'una altra manera, cvHead i ehci respectivament. Observant aquestes dues taules amb cura, ehci mostra un rang màxim d'angles una mica més gran que cvHead. A més a més, en general ehci té menys error en les mostres obtingudes que cvHead. Per tant, fins a aquest punt ehci és millor candidat que cvHead. Mirant internament els resultats de cadascuna de les taules per separat, tant ehci i cvHead en general tenen menys error en les mostres sobre l'eix X que sobre l'eix Y. Això és degut al fet que per ocultar els punts de referència, i per conseqüència perdre'ls, de la fisonomia dels usuaris aplicant una rotació *pitch* és molt complicat per qüestions de límits físics. En canvi, perdre aquests punts en aplicar una rotació *yaw* és molt més senzill, ja que el mateix nas o part de la cara oculta els punts.

Finalment hi ha una clara diferència en els softwares restants. Els límits fins a on fTracker funciona, és a dir, fins al punt on es deforma el model per primer cop, són -60 i 60 graus en els eixos X i Y. Tant pyHead com pyFace tenen el mateix rang límit, -30 i 30 graus en l'eix X i -15 i 15 graus en l'eix Y.

	Eix X (pitch)			Eix Y (yaw)		
Angle esperat	Obtingut	Error º	Error %	Obtingut	Error º	Error %
-60	-47.95º	12.04º	20.07%	-17.07º	42.92º	71.53%
-45	-27.17º	17.82º	38.61%	-15.18º	29.81º	66.26%
-30	-18.65º	11.34º	37.80%	-10.49º	19.50º	65.01%
-15	**-12.94º**	**2.05º**	**13.67%**	-5.65º	9.34º	62.28%
0	**2.25º**	**2.25º**	**-%**	**2.52º**	**2.52º**	**-%**
15	**13.86º**	**1.13º**	**7.56%**	8.75º	6.25º	41.62%
30	18.09º	11.90º	39.69%	20.38º	9.61º	32.03%
45	23.59º	21.40º	47.55%	28.78º	16.21º	36.03%
60	26.66º	33.33º	55.55%	37.30º	22.69º	37.82%

Taula 3: Resultats de M10 i M11 per a cvHead.

	Eix X (pitch)			Eix Y (yaw)		
Angle esperat	Obtingut	Error º	Error %	Obtingut	Error º	Error %
-60	-40.82º	19.17º	31.95%	-30.40º	29.59º	49.32%
-45	-29.11º	15.88º	35.29%	-24.64º	20.35º	45.23%
-30	-18.85º	11.14º	37.13%	-22.62º	7.37º	24.59%
-15	**-13.00º**	**1.99º**	**13.27%**	**-13.64º**	**1.35º**	**9.05%**
0	**1.54º**	**1.54º**	**-%**	**-0.96º**	**0.96º**	**-%**
15	**12.00º**	**2.99º**	**19.99%**	**13.23º**	**1.76º**	**11.74%**
30	**24.14º**	**5.85º**	**19.50%**	23.19º	6.80º	22.68%
45	29.75º	15.24º	33.86%	25.86º	19.13º	42.52%
60	38.05º	21.94º	36.58%	30.74º	29.25º	48.75%

Taula 4: Resultats de M10 i M11 per a ehci.

En addició a la comparació de taules, els angles màxims i mínims queden recollits a la Taula 5, és a dir, els resultats de M8 i M9.

Software	Eix X (pitch)	Eix Y (yaw)
ehci	[-15;30]	[-15;15]
cvHead	[-15;15]	[0;0]
fTracker	[-60;60]	[-60;60]
pyFace	[-30;30]	[-15;15]
pyHead	[-30;30]	[-15;15]

Taula 5: Resultats de M8. Angles màxims dels softwares.

Així doncs, els softwares amb més expectatives en convertir-se en candidat per al seu ús en un prototipus funcional són ehci i fTracker. Però, com que fTracker encara que tingui millors resultats en termes de robustesa no compleix els requeriments màxims envers la limitació hardware, el millor candidat és ehci.

4.3 Prototipus funcional

Un cop definit a partir de l'estudi el candidat òptim a utilitzar per a crear el joc de "On està Wally?", en aquest cas ehci, cal definir les pautes de com fer-ho. L'objectiu del joc és que l'usuari interactuï amb aquest movent el cap i mirant de trobar a Wally. Un cop que l'ha trobat, haurà d'esperar un cert temps mirant-lo perquè es doni com a vàlid que ho ha trobat. El motiu del perquè aquesta restricció és perquè l'usuari podria moure el cap per tota la pantalla esperant al fet que surti el missatge que l'ha trobat.

Per tal d'implementar aquest joc, cal desenvolupar un conjunt de funcionalitats que ampliïn el software. Les funcionalitats són:

- La primera funcionalitat és crear un contenidor on poder emmagatzemar els angles de rotació de l'eix X i l'eix Y. A més a més ha de permetre la comparació entre angles per tal de saber si els angles són els mateixos o no. Però com que és gairebé impossible replicar el mateix conjunt d'angles, en altres paraules, igualar els angles d'on es troba Wally amb els del vector d'atenció de l'usuari, la comparació es fa mitjançant un rang i un *offset*. Aquest rang, un per cada angle, es calcula en base de l'angle de la posició de Wally i l'*offset* utilitzant l'Equació 6.

$$\begin{aligned} range &= [a, b] \\ a &= angle - offset \\ b &= angle + offset \end{aligned} \tag{6}$$

- La segona funcionalitat és crear un contenidor que contingui les principals informacions sobre l'escenari de joc, és a dir, l'angle on es troba Wally, la imatge del joc i la variable *offset* per a calcular el rang dels angles. El càlcul de la posició de Wally es fa aplicant la llei de cosinus [24] tal com es mostra a l'Equació 8 aïllant els valors de l'Equació 7, que és l'original. Aquest càlcul s'ha de repetir dos cops, un per l'angle de l'eix X i un altre per l'angle de l'eix Y.

$$c^2 = a^2 + b^2 - 2ab * cos(\gamma) \tag{7}$$

$$\gamma = arccos(\frac{a^2 + b^2 - c^2}{2ab}) \tag{8}$$

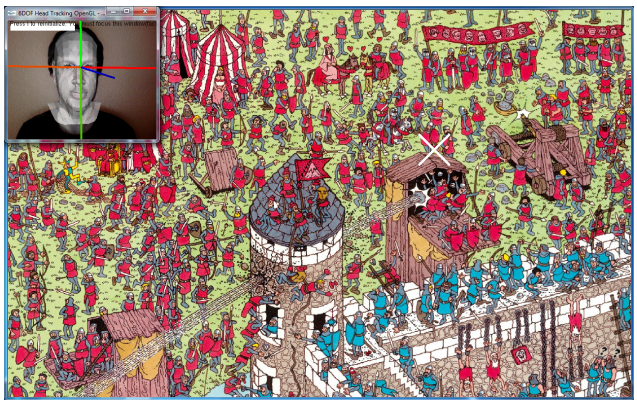


Figura 10: Exemple d'execució del prototipus funcional amb el punter (creu blanca). La finestra del model s'ha deixat a nivell d'exemple, però pot ser minimitzada.

- L'ultima funcionalitat ha de ser l'encarregada de llegir el sistema de fitxers on es guarden les imatges del joc i els diferents angles d'on es troba Wally en cadascun dels casos. D'aquesta forma, el joc és escalable en el sentit de que si s'insereixen noves carpetes amb els fitxers d'informació pertinent, és com afegir un nou nivell de joc.

Un cop les funcionalitats del joc són implementades, es modifica el codi del software per a tal d'integrar-hi aquestes funcionalitats. Per tal de fer-ho, s'afegeix un entorn visual on es mostra la imatge de Wally perquè l'usuari pugui interactuar amb aquesta mitjançant el moviment del cap.

Per tal de facilitar a l'usuari la posició d'on intersecciona el vector d'atenció amb la finestra de joc, s'implementa un punter visual associat a la posició de la mirada. La posició d'aquest es calcula amb la llei de sinus, tal com mostra l'Equació 5 i es pinta a la finestra amb un punt de color blanc.

Un cop l'usuari troba a Wally, aquest ha de ser conscient que ho ha fet. Per facilitar la feina, es mostra un missatge per pantalla per a donar-li aquesta informació i que sigui conscient que ha guanyat.

Ara que l'aplicació està implementada, es pot fer la validació experimental del mètode interaccionant amb el joc fins a trobar a Wally, com mostra la Figura 10.

5 Conclusions

Després del desenvolupament de tot el projecte i analitzar els diferents resultats obtinguts, es pot dir que s'han satisfet els objectius proposats del projecte amb èxit.

No obstant això, després d'experimentar amb els softwares i analitzar els resultats es pot arribar a les següents conclusions:

- Ehci és el candidat que més s'ajusta a les limitacions del dispositiu Raspberry Pi. Cal dir, que té certes limitacions en els angles màxims i mínims, ja que el rang és de [-15;30] en rotació *pitch* i [-15;15] en rotació *yaw*, inclús quan aquest utilitza RANSAC per a millorar el model.

- fTracker és el software amb més robustesa de tot l'estat de l'art i possiblement el que s'hauria d'utilitzar per a una aplicació funcional. Tot així, caldria gaudir-lo de

Pose Estimation i ampliar-ho amb algun mètode per a resoldre les oclusions.

- Tant pyFace com pyHead, tenen problemes tant de rendiment com de robustesa. Cal escalar-los molt perquè siguin igual de controlables que la resta de softwares avaluats en aquest projecte.
- CvHead a pesar de tenir bons resultats en rendiment, cal millorar la seva robustesa en el model per tal que sigui mínimament funcional.
- En general, tots els softwares de l'estat de l'art avaluats tenen greus problemes envers les oclusions.
- Per tal d'arribar a utilitzar alguns dels softwares de l'estat de l'art en una aplicació real, caldria millorar molt la seva robustesa.

5.1 Treball futur

Primerament, la continuació d'aquest projecte ve donada envers estudiar al millor candidat, en aquest cas ehci, en l'aplicació real en un dispositiu tipus Raspberry Pi.

El següent punt a considerar és que existeixen diverses parts millorables per a incrementar tant el rendiment com la robustesa dels softwares. Per a millorar el rendiment caldria modificar el tractament de la memòria, reduint-la i millorant-la, minimitzant el seu consum i els accessos necessaris a aquesta. També intentar no utilitzar VRAM, ja que al final utilitzar-la significa moure part de les pàgines del programa que actualment estan a memòria a disc, ralentitzant l'accés quan es vulgui accedir a aquest conjunt de dades. Altrament, s'hauria de millorar l'eficàcia en rapidesa de l'execució del software per a aconseguir un volum de frames per segon més gran per a tenir un model més estable.

Respecte a millorar la robustesa dels softwares, s'hauria d'estudiar la possibilitat d'afegir algun mètode de Facial Landmark Estimation com ara bé AAM [20], Elastic Graph Matching [21] o Contrained Local Models [22]. Tanmateix, alhora es podria investigar el fet d'afegir algun mètode per predir el moviment i avançar-se a les possibles oclusions del model. Malgrat que són bones idees, segurament empitjorarien el rendiment del software considerablement, així que s'hauria de mirar la viabilitat d'aplicar-los o no, si se suposa que encara s'avaluen respecte de les limitacions del dispositiu.

5.2 Agraeïments

Personalment, m'agradaria donar les gràcies a la meva família i amics per ajudar-me en tot el possible durant el transcurs del projecte. També al meu tutor Fernando Vilariño per tot el suport que m'ha donat setmana rere setmana, tant presencialment com per email. També, donar les gràcies a Onhur Ferhat pel conjunt d'hores que va perdre ajudant-me a fer funcionar correctament tots els diferents softwares.

REFERÈNCIES

[1] J. Saragih, "Non-rigid Face Tracking", In Mastering OpenCV with Practical Computer Vision Projects, PACKT, Oct. 2012.

[2] J. Saragih, "Deformable Face Alignment via Local Measurements and Global Constraints". In M. González Hidalgo, A. Mir Torres, X. Varona Gómez (Eds.), Deformation Models. Springer, Nov. 2012.

[3] S. Lucey, Y. Wang, J. Saragih and J. Cohn, "Non-rigid Face Tracking with Enforced Convexity and Local Appearance Consistency Constraint", International Journal of Image and Vision Computing (IVC), Volume 28, Issue 5, pp. 781-789, May 2010.

[4] J. Saragih, S. Lucey, and J. Cohn, "Face Alignment through Subspace Constrained Mean-Shifts", IEEE International Conference on Computer Vision (ICCV), 2009.

[5] L. Vacchetti, V. Lepetit and P. Fua, "Fusing Online and Offline Information for Stable 3D Tracking in Real-Time", Swiss Federal Institute of Technology (EPFL).

[6] G. Slabaugh, "Computing Euler angles from a rotation matrix". Technical Report, 1999; https://truesculpt.googlecode.com/hg-history/38000e9dfece971460473d5788c235fbbe82f31b/Doc/rotation_matrix_to_euler.pdf.

[7] Mike Day, "Extracting Euler Angles from a Rotation Matrix", Insomniac Games. Technical Report, Juny 2012; https://d3cw3dd2w32x2b.cloudfront.net/wp-content/uploads/2012/07/euler-angles1.pdf.

[8] D. Baggio, D. Millán, N. Mahmood, R. Shilkrot, S. Emami, K. Ievgen, J. Saragih, "Mastering OpenCV with Practical Computer Vision Projects", PACKT Publishing, 2012.

[9] S. Emami, "Mastering OpenCV with Practical Computer Vision Projects - Full Source Code", 2015; https://github.com/MasteringOpenCV/code.

[10] D. Baggio, "6 Degrees of Freedom Head Tracking", Feb. 2010; https://code.google.com/p/ehci/wiki/6dofhead.

[11] D. Baggio, "6 Degrees of Freedom Head Tracking - Source Code", Feb. 2010; https://code.google.com/p/ehci/downloads/detail?name=ehci-0.7-src.zip.

[12] D. Hariharan, "Head Tracking using Python and OpenCV", 2014; http://dhananjaih.weebly.com/headtracking.html.

[13] D. Hariharan, "Head Tracking using Python and OpenCV - Source Code", 2014. https://github.com/DhananjaiH/Head-tracking.

[14] FreeTrack, 2015; http://www.free-track.net/english/.

[15] TRACKIR, 2015; https://www.naturalpoint.com/trackir/.

[16] P. Maciejewski, "cvHeadPose - Source Code", 2011; https://github.com/pwlmaciejewski/cvHeadPose

[17] V. Blanz, T. Vetter, "Face Recognition Based on Fitting a 3D Morphable Model", IEEE TPAMI, Volume 25, No. 9, 2003.

[18] T. Cootes, K. Walker, C. Taylor "View-based Active Appearance Models", In IEEE, 2000.

[19] A. Pentland, B. Moghaddam, T. Starner, O. Oliyide, M. Turk, "View-based and Modular Eigenspaces for Face Recognition", In Computer Vision and Pattern Recognition, IEEE, pp. 84-91, 1994.

[20] T. Cootes, G. Edwards, C. Taylor, "Active Appearance Models", IEEE TPAMI, Volume 2, pp. 484-498, 2001.

[21] T. Leung, M. Burl, P. Perona, "Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching", In ICCV 1995.

[22] D. Cristinacce, T. Cootes, "Feature Detection and Tracking with Constrained Local Models", In BMVC, pp. 929-938, 2006.

[23] "Manifesto for Agile Software Development", 2001; http://agilemanifesto.org/.

[24] D. E. Joyce, "The Laws of Cosines and Sines", Department of Mathematics and Computer Science, Clark University, 1997; http://www.clarku.edu/~djoyce/trig/laws.html

[25] D. Pasta, "Using Statistical Graphics to Understand Your Data (Not just to Present Results)", Ovation Research Group, Technical Report.

[26] E. Tufte, "The Visual Display of Quantitative Information", Graphics Press, 1983.

[27] L. Morency, A. Rahimi, N. Checka, T. Darrel, "Fast Stereo-Based Head Tracking for Interactive Environments", Technical Report, MIT AI Lab, 2002.

[28] K. Hejn, J. Peter Rosenkvist, "Headtracking Using a Wiimote", Department of Computer Science, University of Copenhagen, Technical Report, 2008.

[29] M. Ubilla, D. Mery, R.F. Cadiz, "Head Tracking for 3D Audio Using the Nintendo Wii Remote", Department of Computer Science, Pontificia Universidad Catolica de Chile, 2010.

[30] F. Caballero, I. Maza, R. Molina, D. Esteban, A. Ollero, "A Robust Head Tracking System Based on Monocular Vision and Planar Templates", Sensors (Basel, Switzerland), 9(11), 8924–8943. doi:10.3390/s91108924.

[31] P. Viola, M. J. Jones, "Robust Real-time Face Detection", IJCV, Volume 57, Issue 2, pp. 137-154, 2004

[32] P. Viola, M. J. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features", In Computer Vision and Pattern Recognition, Volume 1, 2001.

[33] NumPy, 2015; http://www.numpy.org/

[34] X. P. Burgos-Artizzu, P. Perona, P. Dollár, "Robust Face Landmark Estimation under Occlusion", In Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV '13), IEEE Computer Society, pp. 1513-1520, DOI=10.1109/ICCV.2013.191.

[35] M. A. Fischler, R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography", SRI International, Volume 4, Issue 6, pp. 381-395, 1981.

[36] R. Raguram, J.M. Frahm, M. Pollefeys, "A comparative analysis of RANSAC techniques leading to adaptive real-time Random Sample Cosensus", Department of Computer Science, University of North Carolina at Chapel Hill and ETH Zürich, 2008.