Universitat de Girona

Departament d'Arquitectura i Tecnologia de Computadors.

Projecte Final de Carrera

Correcció postural esportiva utilitzant un sensor Kinect.

Estudis: Enginyeria tècnica en informàtica de Sistemes.

Autor: Asier Menéndez Corral.

Director: Xavier Cufí Sole.

Girona, 2014

1 Índex

2 Índex de figures

3 Introducció, motivacions, propòsit i objectius del projecte

Les millores tecnològiques al camp del reconeixement visual d'imatges millora dia a dia, fins al punt que una tecnologia que fins ara no estava disponible pel públic general avui podem trobar-la en molts dispositius amb un preu molt accessible; gràcies a això podem utilitzar-les per crear noves aplicacions d'utilitat per la nostra vida diària.

Practicar una activitat esportiva a casa sense la supervisió professional inicial o sense cap tipus de coneixement bàsic sobre aquesta pot provocar lesions de diferent ordre, especialment relacionades amb els tendons del nostre cos.

El propòsit principal d'aquest projecte és mostrar com adaptar aquestes tecnologies, a l'abast de qualsevol particular, de forma que un usuari durant la pràctica d'una activitat esportiva concreta, rebi informació visual continua i en temps real dels moviments i gestos incorrectes que està realitzant, en base a uns paràmetres prèviament establerts.

3.1 Objectius

L'objectiu d'aquest projecte consisteix en fer una lectura constant en temps real d'una persona practicant una selecció de diverses activitats esportives estàtiques utilitzant un sensor Kinect.

A través de les dades obtingudes pel sensor Kinect i utilitzant les llibreries de "skeleton traking" proporcionades per Microsoft s'haurà d'interpretar les dades posturals obtingudes per cada tipus d'esport i indicar visualment i d'una manera intuïtiva els errors que està cometent en temps real, de manera que es vegi clarament a quina part del seu cos realitza un moviment incorrecte per tal de poder corregir-lo ràpidament.

Els esports que s'han seleccionat han sigut les 3 de les activitats esportives més practicades a casa, i que al tenir un impacte important a les articulacions poden provocar lesions si no són realitzades correctament. Aquestes són: Córrer a cinta estàtica, salt amb corda, boxa amb sac.

Per aconseguir aquest objectiu, inicialment es va dividir el PFC en diferents tasques:

- Estudi de la precisió i resolució oferida pel sensor Kinect.
- Selecció d'activitats esportives més adequades, recopilació d'informació de dades posturals correctes.
- Incorporació de les llibreries de "skeleton tracking" de Microsoft Kinect SDK per saber com funciona i quines dades ofereix l'esquelet.
- Incorporació i càlcul de dades necessàries per interpretar les dades llegides pel sensor, especialment els angles a les extremitats del nostre cos.
- Simular per conèixer la precisió i sensibilitat de les dades llegides.
- Crear l'aplicació final amb les correccions sobre precisió realitzades.
- Provar i avaluar l'aplicació final realitzant les proves necessàries.

4 Estudi de viabilitat

Els requisits pel desenvolupament són totalment viables, ja que les eines que s'utilitzen per desenvolupar-lo no necessiten d'un pressupost excessiu ni d'un maquinari de dificil accés. A nivell de desenvolupador, el maquinari consta d'un ordinador personal (els requisits del qual s'especifiquen al capítol 7) amb una potència mitjana, per poder treballar amb comoditat i d'un petit desemborsament de 100€ destinats al dispositiu Kinect de Microsoft, ja que és l'element al voltant del qual es treballa.

Pel que fa al software, totes les llibreries utilitzades, tant les pertanyents al "*Framework*" com les del "*SDK de Kinect*" són lliures. El programa informàtic que s'utilitza per desenvolupar l'aplicació es Microsoft Visual Studio 2010, que sí que requereix una llicència vàlida, però que és gratuïta en la versió "Express" descarregable a través de la pàgina oficial.

Finalment, la càrrega de treball d'aquest projecte, al no ser extremadament ambiciós, pot ser assumida per una sola persona.

5 Metodologia

La metodologia de treball que s'ha seguit en aquest projecte ha sigut en forma seqüencial, dividida en tasques i subtasques amb certa dependència i superposició entre alguna d'elles.

El motiu de triar aquest tipus de metodologia ha sigut que abans de començar a desenvolupar l'aplicació era necessari realitzar uns processos d'aprenentatge i familiarització amb els llenguatges de programació C# i XAML, un estudi de les llibreries i les classes de les llibreries del Microsoft Kinect SDK que utilitzarem i documentació general sobre les activitats tractades.

Les tasques principals han sigut:

- **Aprenentatge:** Aquesta tasca ha sigut necessària per entendre tots els conceptes que suposaven els nous llenguatges i llibreries, així com tornar a familiaritzar-se amb Microsoft Visual Studio. Ja que els nous conceptes, llibreries internes i funcionalitats de C# i XAML es van descobrint al llarg de tot el projecte, és una tasca que abasta quasi tota la duració del projecte. Aquesta tasca consta de les següents subtasques:
    - *Visual Studio 2010: Una familiarització inicial de dues setmanes.*
    - *C#: Els nous coneixements d'aquest llenguatge són necessaris durant tota la duració del projecte.*
    - *XAML: ""*
    - *SDK Kinect 1.8: Un estudi inicial exhaustiu, de vàries setmanes, de totes les possibilitats que ofereix el SDK i quines funcionalitats es poden aprofitar per a aquest projecte.*
- **Investigació:** En aquesta tasca s'ha procedit a documentar-se sobre el funcionament específic del programari i maquinari que s'utilitzarà, així com la cerca d'informació

professional de les activitats que es volen tractar en aquest projecte. Aquesta tasca consta de les següents subtasques:

- *Sensor Kinect: Documentació inicial, limitacions i característiques del sensor.*
- *Precisió Kinect: Un cop completa la subtasca anterior, es comprova la precissió de les mesures de kinect amb el software de calibratge disponible.*
- *Activitats esportives: Recopilació de tota la informació professional d'articles sobre activitats esportives i lesions. Aquesta subtasca s'ha realitzat juntament amb altres, al llarg dels 2 primers mesos.*
- *Llibreries Kinect SDK: Un cop familiaritzat de nou amb Visual Studio 2010, es realitza un anàlisi de menys d'un mes de les llibreries que s'utilitzaran, amb totes les proves necessàries per entendre com funcionen les classes i mètodes.*
- *Llibreries Skeleton Tracking: Es realitza el mateix procés que la tasca anterior, però més exhaustiu, amb les llibreries específiques de "Skeleton Tracking" al llarg de 2 setmanes.*

- **Disseny:** Aquesta tasca contempla el disseny general del projecte. En aquesta fase s'esquematitzen quines funcionalitats es volen a l'aplicació i de quina forma s'aconseguiran un cop s'han adquirit els coneixements de les tasques anteriors. Aquesta tasca dura un mes i mig, i es superposa en gran part amb la següent tasca general (implementació), ja que moltes idees que inicialment semblaven bones sobre el paper, s'han hagut de modificar basant-se en les dades que ens proporciona el sensor. Aquesta tasca consta de les següents subtasques:
  - *Càlcul de posicions: Un cop es té tota la informació sobre les activitats esportives a tractar, es decideix quines operacions es realitzaran per obtenir les dades que ens interessen.*
  - *Disseny general: Idea general de l'aplicació que es desenvoluparà en conceptes generals.*
  - *Disseny aplicació C#: Com s'estructurarà el codi l'arxiu ".cs" per obtenir els resultats generals.*
  - *Disseny UI: Quins elements necessitem per interactuar amb l'usuari de manera que pugui fer servir l'aplicació intuïtivament.*
- **Implementació:** Aquesta part és a la que es realitza tota la feina de programació i implantació. Com és obvi, és la part més extensa del projecte i requereix totes les tasques realitzades anteriorment. Dura un total de 4 mesos i està composta per les següents subtasques:
  - *Captura d' "stream" d'imatge: Aquesta subtasca es realitza conjuntament amb la fase de disseny per conèixer el tipus de dades que obtenim a l'aplicació per part del sensor.*
  - *Reconeixement de punts d'esquelet i mostra per pantalla de l'esquelet: Aquestes dues subtasques es realitzen un cop finalitzada l'anterior, per tal de començar a manipular i mostrar dades d'esquelets.*

    - *Funcions i càlculs: Aquesta subtasca és la més exigent, ja que és aquí on es creen tots els mètodes i funcions necessaris per realitzar els càlculs posturals. S'inicia en finalitzar les dues anteriors.*
    - *Mostra per pantalla de l' "output": Aquesta subtasca es realitza en conjunt amb l'anterior per comprovar que els resultats de les correccions que es mostren per pantalla s'ajusten a l'objectiu buscat.*
- **Avaluació:** Durant aquesta tasca final, es verifiquen i s'ajusten els resultats, fent petites correccions durant la pràctica d'activitats esportives, per tal que l'aplicació respongui de forma ideal.
- **Documentació:** Per últim, a la tasca de documentació, un cop acabat el treball, que consisteix en la preparació i revisió de la memòria, es procedeix a documentar tot el treball fet al llarg del projecte per que aquest quedi plasmat en aquest mateix document.

La càrrega total de treball ha sigut d'unes 360 hores aproximadament, repartides en 100 hores en la tasca d'aprenentatge i investigació, 25 hores de disseny, 200 hores d'implementació, 5 hores d'avaluació i 30 hores de documentació.

Tot i així, en les primeres setmanes del projecte es va utilitzar una metodologia molt menys estructurada, on totes les tasques se superposaven, però a causa del nombre elevat de nous conceptes que s'havien d'assimilar i la falta de temps, com s'explica en el següent capítol, es va decidir ajornar la data d'entrega i optar per una metodologia més estructurada.

6 Planificació

S'inicia aquest PFC al juliol de 2013, amb una planificació incial de 3 mesos, amb expectativa de presentació a la convocatòria de setembre de 2013. El motiu de la decisió de realitzar aquest PFC en un període de temps tan compacte va ser degut a aprofitar un període no laboral per dedicar-li la totalitat del meu temps.

Davant de la impossibilitat de seguir el pla inicial, degut al temps requerit per alguna de les tasques, com familiaritzar-se amb noves eines de treball i llenguatges, aquest ha sigut modificat per una planificació molt més flexible.

De manera que la planificació inicial d'un PFC de menys de 3 mesos ha passat a una de 10 mesos, el resultat del qual ha sigut una aplicació que compleix amb els objectius plantejats.

La planificació queda finalment estructurada en les tasques, subtasques i temps estimat que s'especifiquen al diagrama de Gantt de la següent figura.

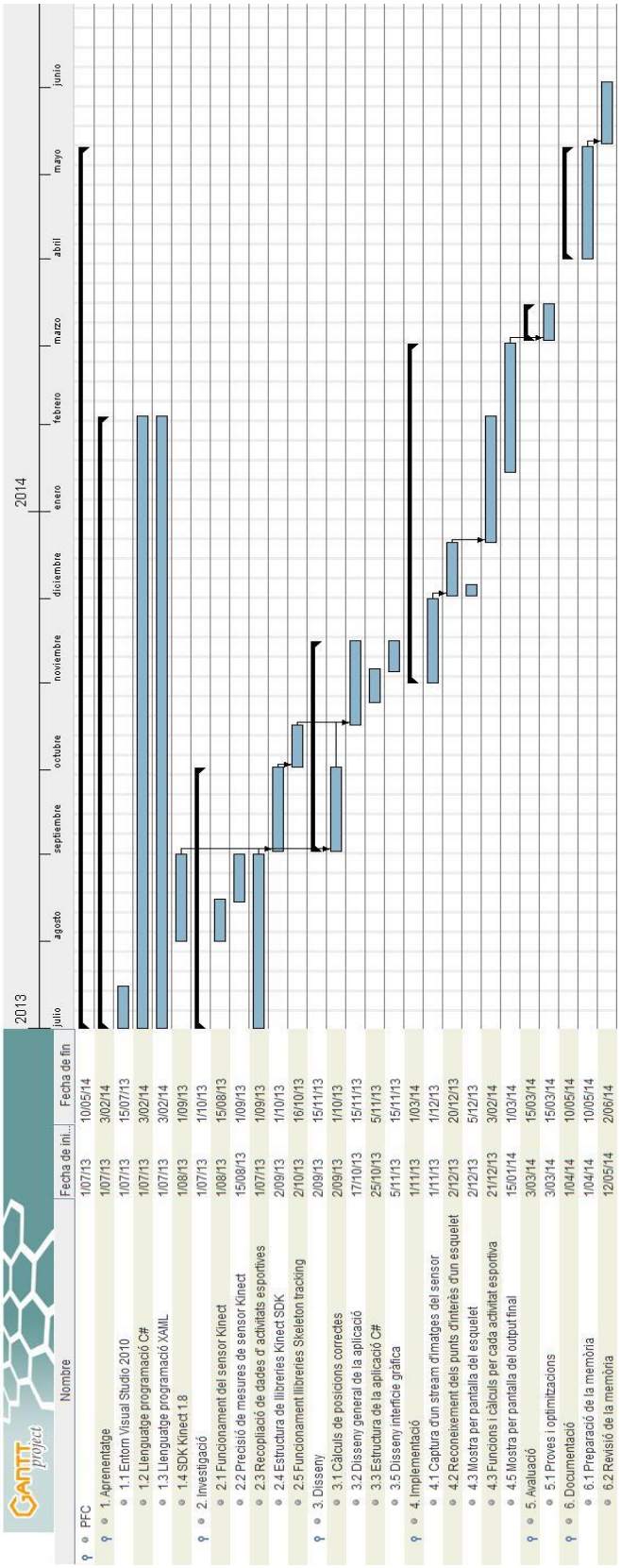

Figura 1: Diagrama de Gantt amb la planificació final del projecte

7 Marc de treball i conceptes previs

7.1 Adquisició de dades de profunditat

Els sistemes d'adquisició de dades de profunditat es poden classificar en 2 categories bàsiques: els que no requereixen contacte físic amb l'objecte a tractar i els que pel contrari, si ho requereixen [6]. Per norma general, els sistemes de contacte directe proporcionen una informació més precisa però requereixen molt més temps per realitzar l'escanejat de l'objecte. A causa de les noves aplicacions trobades i a les millores tecnològiques de visió per computador, el segon tipus va millorant dia a dia.

Dins la categoria de sistemes d'adquisició de dades de profunditat sense contacte físic, existeixen 2 tipus [7].

- **Sistemes sense contacte directe actius**: Emeten algun tipus de radiació o llum i detecten el reflex o la radiació passant a través de l'objecte per analitzar-lo. El tipus d'emissions poden incloure: llum, ultrasons o rajos X.
- **Sistemes sense contacte directe passius:** En comptes d'emetre radiació ells mateixos, detecten la radiació ambiental. La majoria de sistemes d'aquest tipus captura la llum ambiental, encara que també altres tipus de radiació ambiental com els rajos infrarojos poden ser detectats. Per tant, normalment només requereixen una simple càmera digital.

Dins dels tipus de sistemes sense contacte directe actius, que son els rellevants en aquest projecte podem trobar les següents tecnologies:

1. **Mètode Time-of-flight:** Aquests mètodes calculen la distancia que hi ha fins l'objectiu a partir del temps d'anada i tornada d'un pols de llum infraroja. Aquest tipus de mètodes son els més antics, tot i així ofereixen una gran precisió.

2. **Mètode de triangulació:** En aquest cas un emissor emet un làser al objecte a tractar i una càmera observa la localització del punt on està el laser. Depenent de la distancia a la que aparegut el laser, el punt apareixerà a diferents localitzacions del camp de visió de la càmera. Aquesta tècnica s'anomena triangulació perquè utilitza l'emissor, el punt del làser i la càmera com un triangle, del que obtindrem la localització del punt del làser.

3. **Mètode de holografia conoscòpica:** Em aquest mètode, un raig de llum làser es projectat en una superfície o objecte i la reflexió d'aquest amb el mateix patró del raig travessen un cristall conoscòpic, i finalment son projectats en un dispositiu CCD (charged-coupled device). El resultat es un patró de difracció que pot ser analitzat per determinar la distancia fins a la superfície mesurada.

4. **Mètode de llum estructurada:** Les càmeres que utilitzen el mètode de llum estructurada contenen un emissor i un sensor d'infrarojos al costat d'una càmera RGB (com el sistema Kinect utilitzat en aquest projecte). El emissor emet un patró de llum i els sensors detecten

les deformacions obtingudes en la reflexió de tot el camp de visió de la càmera.

5. **Mètode de llum modulada:** Els sistemes que utilitzen aquest mètode emeten una llum que varia constantment cap a l'objecte a analitzar. Normalment la llum canvia en cicles d'amplitud seguint un patró sinusoidal. Una càmera detecta la llum reflectida i la variació del patró determina la distancia que ha viatjat la llum. El sistema de llum modulada també permet al sistema ignorar les llums d'altres fonts diferents que el làser emès.

7.2 Maquinari de visió per computador rellevant

Existeixen molts tipus de sistemes de visió per computador, tot i així tots contenen els següents elements bàsics: font d'energia, un sistema d'aquisició com a mínim (càmera, ccd, etc...), un processador, així com sistemes de control i cables de comunicació o algun tipus de interconnexió inalàmbrica. A més, els sistemes de visió per computador pràctics contenen eines de software per facilitar el desenvolupament d'aplicacions, així com un display per mostrar el que està fent el sistema. Els sistemes de visió per espais industrials contenen, a més, un sistema de il·luminació. Un sistema complert pot incloure també molts d'accessoris com suports de càmeres, cables i connectors.

Degut que aquest projecte es centra en un sistema de visió per computador a l'abast de particulars i no en sistemes pensats per entorns industrials (tals com el control de qualitat) o altament especialitzats (com els utilitzats en la captura gestual per pel·lícules o videojocs), s'enumeraran a continuació els dispositius d'aquest tipus disponibles en el mercat avui en dia, amb les especificacions més remarcables.

Microsoft Kinect for Windows [3]

Figura 2: Sensor Kinect

Consum: 2.25W
Camp de visió: 57° H, 43° V (Horitzontal, Vertical)
Resolució: RGB (1280x960)
Càmera infraroja (sensor de profunditat): VGA (640x480):30fps.
Distància d'operació del sensor de profunditat: 0.4m a 8m

Software: Kinect for Windows SDK 1.8
Preu: 100 Euros.

Asus Xtion Pro Live [4]

Figura 3: Sensor Asus Xtion Live

Consum: 2.5W
Camp de visió: 58° H, 45° V (Horitzontal, Vertical, Diagonal)
Resolució: SXGA (1280*1024)
Càmera infraroja (sensor de profunditat): VGA (640x480): 30 fps, QVGA (320x240): 60 fps
Distància d'operació del sensor de profunditat: 0.5m a 9.2m
Software: software development kits(OpenNI SDK bundled)
Preu: 160 Euros.

***PrimeSense Carmine 1.08* [5]**

Figura 4: Sensor PrimeSense Carmine

Consum: 2.25W
Camp de visió: 57.5° H, 48° V (Horitzontal, Vertical)
Resolució: RGB (640x480)
Càmera infraroja (sensor de profunditat): VGA (640x480): 60fps.
Distància d'operació del sensor de profunditat: 0.25m a 3.5m

Software: NiTe middleware (OpenNI SDK bundled)
Preu: 220 Euros.

7.3 Característiques de activitats esportives tractades

En aquest apartat es descriuen quines activitats esportives s'han triat per realitzar el seu seguiment en l'aplicació, així com els motius i les peculiaritats d'aquestes.

Ja que el dispositiu Kinect necessita d'un entorn estàtic, és imprescindible que l'activitat desenvolupada no requereixi desplaçar-se per grans distàncies i a més, és ideal que pugui ser practicada a interiors normalment, ja que els sensor Kinect funciona d'una manera molt més precisa en espais tancats que oberts.

Per triar les activitats esportives a realitzar el seguiment s'han seguit una serie de factors, que ordenarem de major a menor importància.

1. Pot ser practicada estàticament.
2. Pot ser practicada a interiors.
3. Requereix de moviments continuats de tot el cos.
4. Pot ocasionar lesions de diferent ordre si no es practicada correctament.
5. És practicada per un conjunt gran de població.

A partir d'aquests factors s'han seleccionat 3 activitats esportives que s'adeqüen als objectius d'aquest projecte. Aquestes seran: Running sobre cinta, Salt amb corda i Boxa en punching bag.

Totes són activitats d'entrenament per diferents esports, que compleixen els requisits buscats i al ser 3 activitats amb un impacte gran sobre les nostres articulacions poden ocasionar lesions si es realitzen incorrectament.

Algunes de les activitats que compleixen els 2 primers factors descartades són: Bicicleta el·líptica (no hi ha impacte sobre articulacions), musculació/peses (no implica tot el cos), ioga (no hi ha impacte sobre articulacions i existeixen moltes variacions), etc…

Les lesions comuns que es poden ocasionar per una pràctica incorrecta d'aquestes activitats, amb alguna de les causes i exemples que les poden provocar [14], i que intentarem evitar realitzant un seguiment continuu amb aquesta aplicació son les següents:

- **Contractura muscular:** Mala postura. *Exemple: Córrer sense l'esquena completament recta (running).*
- **Elongació o tiró:** Exercici o moviment brusc. *Exemple : Estirament total del braç durant un cop de puny (boxa).*
- **Ruptura de fibres:** Contracció muscular violenta, Impacte o col·lisió d'un objecte provocant una compressió violenta del múscul contra l'os. *Exemple: Cop de puny al sac de boxa sense la mà completament alineada amb el braç (boxa).*

- **Tendinitis:** Sobrecàrrega muscular. *Exemple: Realitzar el moviment de rotació de la corda implicant les espatlles (salt amb corda).*
- **Perostitis tibial:** Mala tècnica de carrera i anomalies biomecàniques. *Exemple: Avançar la part baixa de la cama per davant del tronc durant la fase de recuperació (running).*

7.3.1 Running sobre cinta

El running (carrera a peu en anglès) és un procés complex i coordinat que involucra tot el cos. Cada ésser humà corre d'una manera diferent, però hi ha aspectes generals de la carrera comuns que s'han d'assolir per realitzar una pràctica eficient i no lesiva.

A més dels obvis moviments de la part inferior del cos d'impuls i recuperació, els moviments de la part superior son molt importants per la carrera, ja que compensen els moviments de la part inferior mantenint el cos en equilibri de rotació. El moviment de recuperació d'una cama es compensa amb el moviment cap a baix del braç oposat, i els moviments de suport i impuls són equilibrats elevant el braç oposat. Les espatlles i tors també s'involucren, ja que l'impuls de la cama és més lent que el cop de peu de recuperació, i el moviment cap amunt del braç també ho és. El moviment cap abaix del braç és més ràpid i fort.

Com menys eficients siguin els moviments de la part inferior del cos, més exagerats seran els moviments de la part superior per absorbir l'impuls.

La major part de l'energia gastada durant la carrera és per equilibrar els moviments. De tal manera s'obtindran grans augments de velocitat i economia eliminant moviments incorrectes i malgastadors.

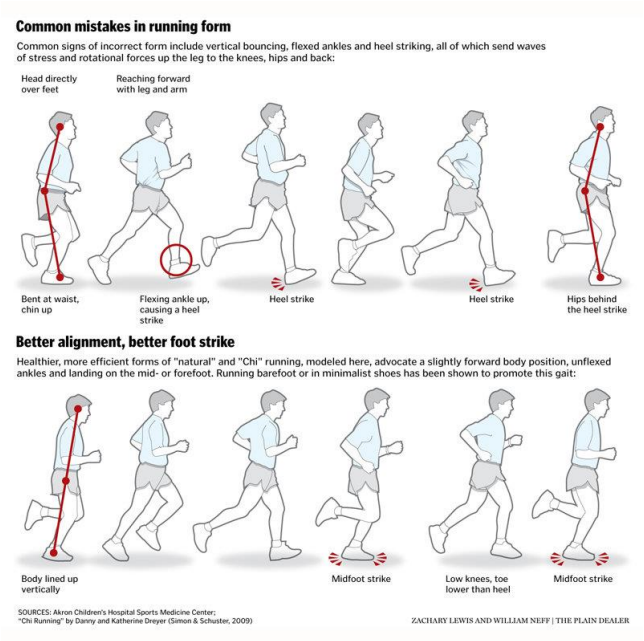

Figura 5: Errors comuns en part inferior del cos durant la carrera

Normalment s'afirma que córrer a cinta produeix més lesions que fer-ho al carrer, encara que no hi han estudis que afirmin aquesta frase. Córrer en cinta, en principi, no es sinònim de lesionar-se més.

Els esportistes professionals aprofiten aquest tipus d'entrenaments per dies on la meteorologia no acompanyi, o per millorar la seva tècnica de carrera davant d'un mirall o una càmera per corregir errors. Aquest és un escenari ideal per l'aplicació desenvolupada.

Per tenir una tècnica correcta de carrera, per tant, caldrà seguir una sèrie de pautes, recollides a partir de diferents articles i webs [12,13] a cada part del cos que s'enumeren a continuació:

- Cap sempre mirant al front, no baixar la mirada mentre es corre.
- Espatlles relaxades, no alçar ni contraure.
- Braços balancejats cap a davant i darrere, no crear-los per davant del pit.
- Colzes situats entre el tors i la cintura, formant un angle aproximat de 90º.
- Tors totalment estirat, evitant ajupir-se.
- Genolls poc elevats i lleugerament flexionats a la fase de suport.
- No estendre mai la cama baixa per davant del cos per evitar que ens faci de "fre" a la fase de suport.
- Peus trepitgen amb la part mitjana de la planta, amb un impacte lleuger.

7.3.2 Salt amb corda

El salt amb corda és un dels millors exercicis cardiovasculars i cremadors de greixos. Quasi cada esport major, tots els relacionats amb arts marcials per exemple, l'utilitza com una part integral del seu entrenament a causa de l'habilitat per millorar l'agilitat i la velocitat dels peus, a més, únicament requereix d'una corda com a material i pot ser practicat a qualsevol lloc.

El motiu per qual la gent no el practiqui més habitualment és degut a la falta de confiança i desconeixement sobre com realitzar aquesta activitat correctament.[15]

Els errors més comuns al practicar aquesta activitat esportiva estan relacionats amb la posició del cos per saltar. Per saltar correctament, s' hauria d'aterrar sempre amb la bola del peu i els genolls han d'estar lleugerament doblegats al aterrar per no carregar-se. Quan s'aterra, els talons no han de tocar mai el terra, ja que fer-ho és molt perjudicial per aquests i provoca que l'esgotament arribi molt abans.

Pel que fa a la part superior del cos, és important mantenir la mirada fixa endavant i no mirant els peus, amb l'esquena recta, intentant relaxar-se. Els encarregats de transmetre el moviment de rotació a la corda seran els canells amb una lleugera ajuda dels colzes, mentre que les espatlles es quedaran fixes a una distància d'uns 15-30º de la vertical, ja que no fer-ho pot provocar un desgast dels músculs i tendons dels manegots dels rotatoris (Rotator cuff's en anglès) de l'articulació i un esgotament molt més elevat.

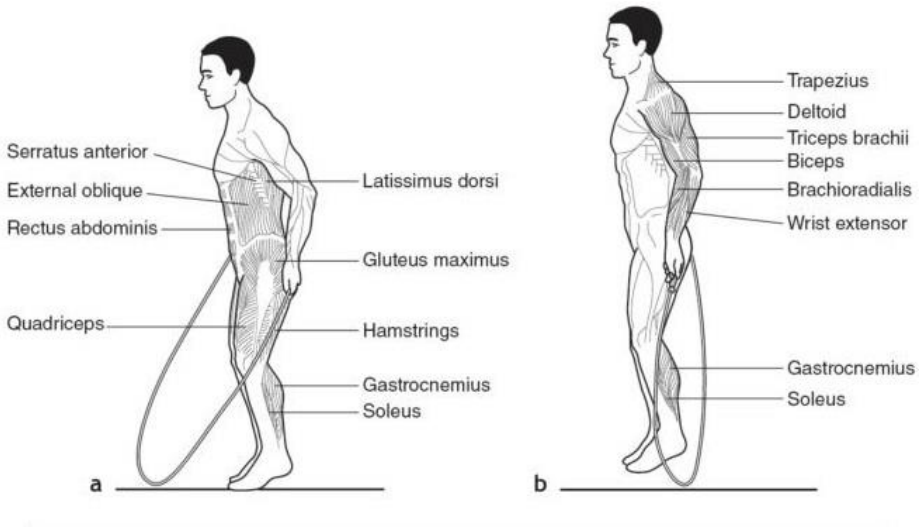

Figura 6: Postura correcta del cos durant l'execució de l'exercici

7.3.3 Boxa amb sac

La boxa (de l'anglès boxing), és un esport de combat a on dos contrincants lluiten utilitzant únicament els seus punys amb guants, i colpegen al seu adversari de cintura cap amunt, a dins d'un quadrilàter dissenyat especialment per aquesta fi. Les lluites estan dividides en diferents assalts o «rounds» d'acord a un reglament que regula les categories de pes i duració del combat.

El sac de boxa (punching bag) és una de les peces d'equipament d'entrenament més antigues i fàcilment reconeixibles. És un sac farcit fet de cuir o plàstic que pot pesar entre 20 a 45 quilos normalment, i és utilitzat per tots els boxejadors per millorar la seva força de cop i la tècnica de boxa. Picar al sac de boxa és una tasca físicament exigent que millora la resistència, força, coordinació i tècnica.

D'entre les activitats que es tracten a aquest projecte, és la més exigent tant físicament com pel que fa a una tècnica correcta, al ser una activitat a la que s'utilitza una gran quantitat de força durant els entrenaments, all llarg d'un temps prolongat. Els requisits mínims per practicar-la són uns guants d'entrenament (lleugerament més pesats que els de competició, +12oz), benes per les mans (per evitar lesions als tendons dels dits i canells) i el sac de boxa.

Ja que a la boxa, a difèrencia de les altres 2 activitats tractades, es realitza un moviment molt més dinàmic i poc constant, en comptes d'analitzar un mateix cicle repetitiu haurem de corregir errors tenint en compte més factors.

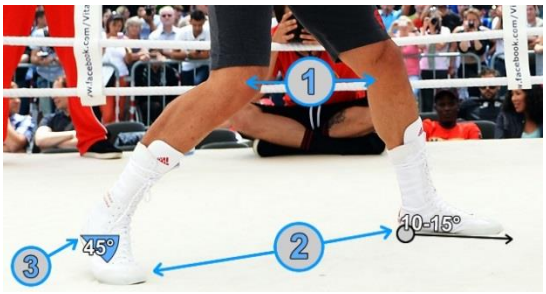

Figura 7: Postura correcta dels peus en guàrdia de boxa

Dins de la boxa existeixen molts estils i tècniques, però hi han unes pautes que s'han de seguir per evitar lesions durant l'entrenament i també per millorar la tècnica de combat [16]. Utilitzant el sensor Kinect es corregiran els errors que sigui capaç de detectar, però d'altres, com per exemple mantenir la vista sempre on es vol picar, seran impossibles de reconèixer. Per tant, dins d'aquestes dues categories de pautes detectables es classifiquen aquests moviments en:

Tècnica de combat:

- Quan es llença un cop de puny l'altra mà ha d'estar sempre en guàrdia, protegint el cap per evitar nockejos.
- El peu enrederit no pot estar mai pla al terra abans de llençar un cop, ja que és l'encarregat de transmetre tota la força del cos, des del terra al puny (peu dret per dretans).
- La cama avançada (cama esquerra per dretans) ha d'estar sempre lleugerament doblegada per mantenir la nostra mobilitat.
- Si no estem llençant cops amb els punys, els peus han d'estar en moviment, intentant-nos desplaçar amb les puntes dels peus.

Per evitar lesions:

- El puny tancat ha d'estar sempre alineat amb el braç, ja que en el cas contrari, al colpejar fortament al sac, es poden provocar lesions molt greus que requereixen mesos de recuperació.
- El braç no pot quedar completament estirat en llençar un cop de puny, ja que si fallem i no toquem al nostre objectiu es pot produir una hiperextensió del tendó del colze.

8 Requisits del sistema

Per poder desenvolupar l'aplicació utilitzant el SDK de Kinect en un ordinador personal és necessari complir una sèrie de requisits que es detallen a continuació.

Sistemes operatius i arquitectures vàlids:

- Windows 7
- Windows 8

- Windows Embedded Standard 7
- Windows Embedded Standard 8

Requisits de hardware mínims:

- L'ordinador personal ha de complir els següents requisits mínims.
  - Processadors 32-bit (x86) o 64-bit (x64)
  - Processador Dual-core, 2.66-GHz o més ràpid
  - USB 2.0 dedicat al dispositiu Kinect
  - 2 GB de RAM
  - Targeta gràfica que suporti DirectX 9.0c
- Un dispositiu "Microsoft Kinect for Windows".

Requisits de software mínims per desenvolupar l'aplicació:

- Visual Studio 2010 o Visual Studio 2012. Les edicions Express gratuïtes poden ser descarregades de la web de Microsoft.
- .NET Framework 4 (instal·lat en Visual Studio 2010), o .NET Framework 4.5 (instal·lat amb Visual Studio 2012).

9 Estudis i decisions

9.1 Maquinari utilitzat

9.1.1 Càmera RGB-Depth Kinect

El sensor Kinect és el dispositiu principal d'aquest projecte, que es basa en les característiques i eines d'aquest per portar-se a terme. El motiu de triar aquest dispositiu entre les altres opcions presentades a apartats anteriors són les següents:

- És el dispositiu amb una comunitat més gran de desenvolupadors al darrere, fins al punt que abans que sortís el SDK per Kinect oficial de Microsoft, van sortir unes llibreries i APIs de OpenNI per fer possible la creació d'aplicacions utilitzant el sensor a qualsevol ordinador personal.
- És el primer i més comercial dispositiu d'aquest tipus, que s'ha anat millorant en cada versió.
- El seu preu en comparació a altres dispositius similars és el més econòmic.
- És un dispositiu que ja es troba present a moltes llars que tenen la consola de videojocs Xbox360, fent innecessària la seva compra de nou.

Aquest dispositiu va ser creat inicialment per Microsoft per a la seva consola de videojocs Xbox 360. Kinect permet controlar i interactuar amb la consola sense necessitat de tenir contacte físic amb un controlador de videojocs clàssic, mitjançant reconeixement de gestos, imatge i veu. Ja que inicialment només podia ser utilitzat amb la consola Xbox 360, es va haver d'utilitzar enginyeria

inversa per obtenir la informació que contenia. Amb aquest procés es va començar a utilitzar el dispositiu a ordinadors en diferents aplicacions com el reconeixement d'objectes.

Finalment, al veure l'interès de la comunitat en utilitzar les característiques d'aquests dispositius en aplicacions d'ordinador personal, Microsoft va llençar al mercat el dispositiu "Microsoft Kinect for Windows" que inclou un SDK, un connector USB i un adaptador per un port de corrent.

9.1.1.1 Interfície de connexió

El "port Kinect" és un connector propietat de Microsoft que proporciona un flux de corrent i de dades al sensor Kinect. La idea inicial de Microsoft era utilitzar un port USB, però a causa de la necessitat de energia addicional a causa del consum del motor pivot de la base, que permet un moviment de -27º a 27º verticals, un port USB no era suficient. De manera que pels models antics de Xbox 360, i per ordinadors personals, s'utilitza un port USB a més d'un adaptador de corrent addicional per proporcionar l'energia necessària. Per contra, les versions més noves de Xbox 360 ja inclouen un port de Kinect dedicat, que estalvia la necessitat d'utilitzar l'adaptador de corrent.

Figura 8: Connector USB i adaptador de corrent Kinect

9.1.1.2 Especificacions de la càmera RGB

La càmera de Kinect és un dispositiu RGB amb una resolució de 640x480 i 24 bits de rang de color (canals vermell, verd i blau).

És un càmera que treballa a 30 frames per segon, com moltes de les que es troben a diferents dispositius que hi ha al mercat d'una forma molt habitual. La càmera és capaç de capturar a una resolució de 1280x960 però els FPS màxims a aquesta resolució estan al voltant de 15. En disminuir la resolució, els FPS augmenten proporcionalment.
El sensor Kinect té una distancia limitada mínima i màxima de l'objecte tractat a la que pot funcionar idealment. Aquesta es troba entre el rang dels 1.2 i 4.5 metres. El camp de visió horitzontal és 57º d'amplitud, el que significa que pot escanejar una escena de 3.8 metres d'amplitud.

El sensor té un camp de visió vertical de 43º. El camp de visió pot ser millorat amb el sistema de pivotatge vertical que permet augmentar-lo o disminuir-lo en 27º en qualsevol direcció vertical.

Figura 9: Càmera RGB del sensor Kinect

9.1.1.3 Especificacions del sensor i l'emissor de profunditat IR

Un emissor de senyals infraroges (IR) i un sensor de profunditat IR permeten al dispositiu Kinect la capacitat de fer mesures de profunditat. L'emissor emet feixos de llum infrarojos i el sensor llegeix els feixos reflectits. Aquests feixos de llum són convertits en informació de profunditat, calculant la distància des de l'objecte al sensor. Tot això fa possible capturar una imatge de profunditat.

El sensor de profunditat està composat per un projector IR i un sensor CMOS monocrom. Localitzat al costat de la càmera RGB, el sensor de profunditat té una resolució de 640x480 amb 16 bits de sensibilitat.

Això significa que pot veure aproximadament 65.000 variacions de gris. Com la càmera RGB, el sensor també captura vídeo a 30fps, encara que depenent de la potència de la màquina on estigui connectada o la complexitat de la tasca a realitzar, els frames per segon poden disminuir [8].

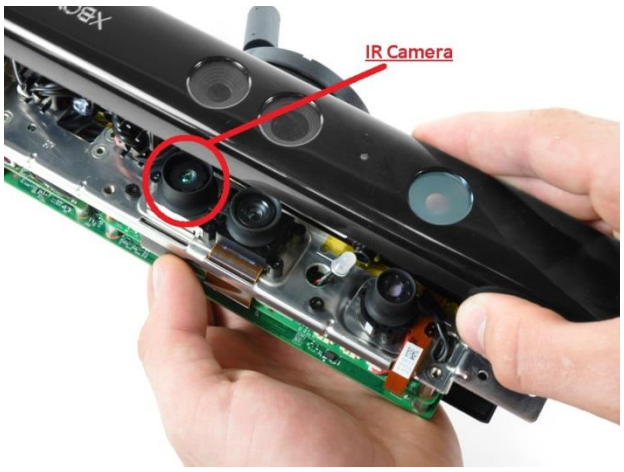

Figura 10: Sensor de profunditat IR de Kinect

A continuació s'enumeren les característiques tècniques del sensor IR:

- **Rang:** de 50cm a 4.5 metres. Es pot modificar el mínim (40cm) i màxim (3m) per aplicacions específiques que requereixin estar a prop del dispositiu.
- **Resolució Horitzontal:** 640x480 i 45 graus de camp de visió vertical, i 58 graus de camp de visió horitzontal (FOV). La geometria mostra que això correspon a uns 0.75mm per píxel als eixos X i Y a una distància de 50cm del sensor, i uns 3mm per píxel a una distància de 2m del sensor.[8]
- **Resolució de profunditat:** ~ 1.5 mm a 50cm i 5cm a 5m [17].
- **Soroll:** +-1mm a distàncies curtes i +-5cm a distàncies llargues.

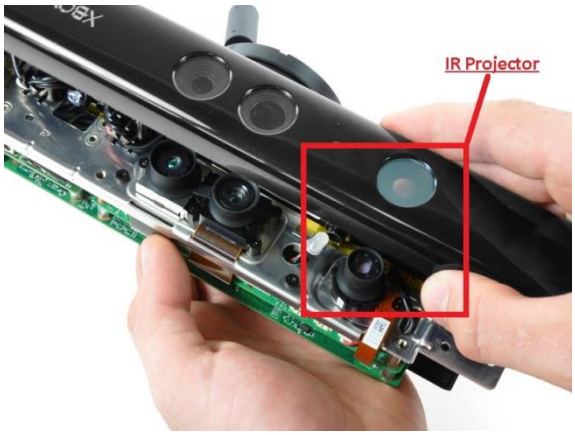

Figura 11: Emissor de senyals IR del sensor Kinect.

9.1.1.4 Especificacions del set de micròfons

El micròfon del dispositiu Kinect és capaç de localitzar les fonts de senyal acústiques i suprimir el soroll ambiental. Aquestes característiques van ser pensades per permetre als usuaris xatejar a través de l'aplicació "Xbox Live" sense necessitat d'utilitzar uns cascos amb micròfon.

Físicament, el micròfon està format per un conjunt de 4 micròfons encapsulats. Cadascun dels 4 canals processa àudio a 16 bits a una freqüència de 16KHZ. Aquest sistema permet aïllar la veu dels usuaris del soroll de l'habitació encara que estiguin allunyats de la consola.

Figura 12: Set de 4 micròfons del sensor Kinect

9.1.2 Ordinador personal

L'altre element imprescindible utilitzat en aquest projecte és un ordinador personal. Primerament com a eina de treball per desenvolupar l'aplicació, i segon, a causa de l'elevada quantitat d'informació a tractar durant l'execució de l'aplicació. Un ordinador més potent serà capaç de donar una sortida d'imatges per segon més elevada, tal com s'ha explicat anteriorment.

Aquest element s'encarregarà de rebre les dades del sensor Kinect a través d'un dels ports USB, interpretar-la, tractar-la i mostrar per pantalla la informació a l'usuari en temps real, que podrà visualitzar-la a través del monitor o utilitzant un altre aparell de visualització gràcies als ports VGA o HDMI.

Figura 13: Model d'ordinador personal utilitzat al projecte

El model d'ordinador personal utilitzat és un ASUS N53SV amb les següents característiques tècniques principals:

- Processador i5-2430M 2.4GHZ
- Memòria ram 4GB
- Targeta gràfica NVidia GT540M

9.2 Programari i llenguatges utilitzats

En aquesta secció es donaran detalls del software utilitzat en aquest projecte. Això te una importància considerable si es vol desenvolupar un projecte similar al tractat o ampliar el mateix. Cal tenir en compte que en aquest projecte s'han utilitzat un nombre considerable de llibreries de lliure distribució, pel que és necessari que les versions utilitzades siguin iguals o superiors a les descrites en aquesta memòria.

Finalment, és important tenir també en compte el Sistema Operatiu utilitzat per desenvolupar aquest projecte, que en aquest cas ha sigut Windows de Microsoft, concretament la versió Windows 7 Professional.

9.2.1 Microsoft Visual Studio 2010

Microsoft Visual Studio és un entorn de desenvolupament integrat (IDE) per sistemes operatius Windows . Suporta múltiples llenguatges de programació com C++, C# (l'utilitzat en aquest projecte), Visual Basic .NET, F#, Java, Python, Ruby, PHP; així com entorns de desenvolupament web com ASP.NET MVC, Django, etc...[10]

Visual Studio permet als desenvolupadors crear aplicacions per Microsoft Windows, aplicacions web i serveis web en qualsevol entorn que suporti la plataforma .NET (a partir de la versió .NET 2002).

El llenguatge amb el qual es treballarà sobre Microsoft Visual Studio 2010 és C#. Aquesta decisió ha sigut pressa pel fet que dins dels llenguatges de programació possibles a utilitzar (C++, C# i Java) per interactuar amb les llibreries de "Kinect for Windows SDK 1.8", la part destinada a C# és la més actualitzada i que per tant presenta més funcionalitats, a més de tenir la comunitat més gran de desenvolupadors d'entre les 3 mencionades.

és al que les llibreries de Microsoft Kinect SDK estan més desenvolupades i presenten més funcionalitats.

El llenguatge utilitzat per realitzar la interfície d'usuari de l'aplicació ha sigut XAML, que permet mostrar totes les dades visuals rellevants, així com interactuar amb l'usuari.

9.2.1.1 C#

C# (C Sharp) és un llenguatge de programació orientat a objectes desenvolupat i estandarditzat per Microsoft com a part de la seva plataforma .NET, que va a ser aprovat com estàndard per la ECMA (ECMA-334) i ISO (ISO/IEC 23270). C# és uns dels llenguatges de programació dissenyats per a la infraestructura de llenguatge comú.

La seva sintaxi bàsica deriva de C/C++ i utilitza el model d'objectes de la plataforma .NET, similar a la de Java, encara que inclou millores derivades d'altres llenguatges.

Encara que C# forma part de la plataforma .NET, aquesta última és una API (Interfície de Programació d'Aplicacions), tals com ho són OpenGL, OpenCL, Win32, etc...; mentre que C# és un llenguatge de programació independent (tal com poden ser-ho C++, Java, PHP, etc...), dissenyat per generar programes a sobre d'aquesta plataforma, la qual utilitzarà com a capa d'abstracció. Existeix un compilador implementat, que proveeix al marc Mono-DotGNU, el qual genera programes per diferents plataformes com Windows, Unix, Andoid, iOS, Windows Phone, Mac OS i GNU/Linux.

9.2.1.2 XAML

XAML (acrònim de eXtensible Application Markup Language) és el llenguatge de format per a la interfície d'usuari per la Base de Presentació de Windows (WPF per les seves sigles en anglès) o Silverlight, el qual és un dels pilars de les interfícies de programació d'aplicacions .NET a la versió 3.0.

XAML és un llenguatge declaratiu basat en XML (eXtensible Markup Language), optimitzat per descriure gràficament interfícies d'usuaris visuals riques, des del punt de vista gràfic, tals com les creades utilitzant Adobe Flash .

En el seu ús típic, els arxius de tipus XAML són produïts per una eina de disseny visual, com la utilitzada en aquest projecte (Microsoft Visual Studio). El XML resultant és interpretat instantàniament per un subsistema de Windows (a partir de la Versió Windows Vista), que reemplaça al GDI (Graphic Device Interface) de les versions anteriors de Windows. Els elements de XAML s'interconnecten amb objectes del "Entorn comú d'execució per llenguatges". Els atributs es connecten amb propietats o esdeveniments d'aquests objectes.

XAML va ser dissenyat per suportar classes i mètodes de la plataforma .NET que tenen relació amb la interacció amb l'usuari, en especial al desplegament en pantalla.

9.2.2 Kinect for Windows SDK 1.8

"Kinect for Windows SDK" proporciona les eines i APIs necessàries per desenvolupar aplicacions amb el dispositiu Kinect per al sistema operatiu Microsoft Windows. Desenvolupar aplicacions per al dispositiu Kinect és essencialment el mateix que desenvolupar qualsevol altra aplicació de Windows, exceptuant que Kinect SDK proporciona suport per funcionalitats de Kinect, com rebre imatges en color, de profunditat, entrada d'àudio i "dades d'esquelet".

Cal destacar que encara que aquest SDK és exclusiu per Sistemes Operatius Windows, existeixen unes llibreries de lliure distribució no oficials anomenades "OpenKinect", que encara que no tenen les mateixes funcionalitats, permeten el desenvolupament, d'aplicacions que utilitzin el mateix dispositiu, en Sistemes Operatius LINUX i MAC. La comunitat de "OpenKinect" consta d'uns 2000 contribuïdors i la seva web oficial és "www.openkinect.org".

El SDK inclou:

- Controladors i informació tècnica per implementar aplicacions Kinect utilitzant un dispositiu Kinect for Windows.
- APIs de referència i documentació per programar en C#, C++ i Visual Basic. Les APIs proporcionen múltiples fluxos de dades a través de diverses variables de vídeo, CPU i só amb la mínima latència de software possible.
- Exemples per demostrar bones pràctiques de programació utilitzant un sensor kinect.

La Natural User Interface (NUI) és el nucli de la API de Kinect for Windows. A través d'ella es pot accedir a les dades dels següents sensors a dins de les aplicacions desenvolupades:

- Dades d'àudio proporcionades pels micròfons del dispositiu Kinect.
- Dades d'imatges en color i dades d'imatges de profunditat proporcionades per la càmera RGB i el sensor IR.

A més de les capacitats del hardware ja descrites, el software de Kinect té aquestes millores:

- Capacitat de reconèixer i rastrejar un cos humà. Aquesta rutina converteix la informació de profunditat del sensor IR en les diferents articulacions del esquelet del cos humà. Això fa possible rastrejar fins a 2 persones davant de la càmera.
- Integració amb les APIs de Microsoft Speech que permeten implementar un motor de reconeixement de veu a dins de l'aplicació.
- Possibilitat de rastrejar cares humanes amb el SDK Face Tracking.

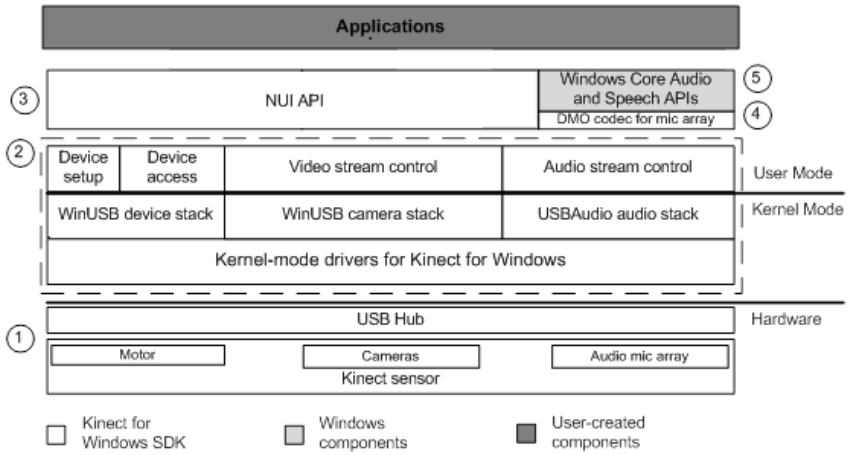

Figura 14: Arquitectura del SDK

Dins d'aquesta API, per tant, es troben incloses llibreries (en el seu nom original) de:

- **Data streaming:** "Audio Stream", "Color Stream", "Depth Stream", "Infrared Stream".
- **Skeletal tracking:** "Tracking Skeletons", "Tracking Modes", "Joint Filtering", "Joint Orientation", "Skeleton Tracking with multiple kinect sensors".
- **Speech**

9.2.2.1 Transmissió de dades: Flux de dades de profunditat

Ja que les dades de profunditat proporcionades per aquesta part de la API del SDK de Kinect són essencials per l'aplicació es farà una descripció d'aquesta.

Cada frame de la informació de profunditat està composta per píxels que contenen la distància (en mil·límetres) que hi ha des de la càmera fins a l'objecte més proper. Una aplicació pot utilitzar

aquestes dades per seguir les dades d'una persona, un objecte, o ignorar objectes que es trobin a una determinada distancia.

El flux d'informació de dades de profunditat conté 2 tipus de dades separades dins d'una estructura de 32 bits:

- Dada de profunditat, en mil·límetres. Representada per un camp de 16 bits.
- Dades de segmentació d'usuari. Cada valor de segmentació de l'usuari és un enter que indica l'índex d'un usuari únic detectat a l'escena. Representat per un camp de 16 bits.

La dada de profunditat és la distància que hi ha fins a un objecte en una coordinada (x,y) en particular, dins del camp de visió del sensor. La profunditat de la imatge està disponible en 3 resolucions diferents: 640x480 (default), 320x240 i 80x60 que s'espicificaran utilitzant el "DepthImageFormat Enumeration". Les opcions de rang s'especificaran utilitzant el "DepthRange Enumeration" que determinarà la distància a partir de la qual el sensor rebrà dades de profunditat.

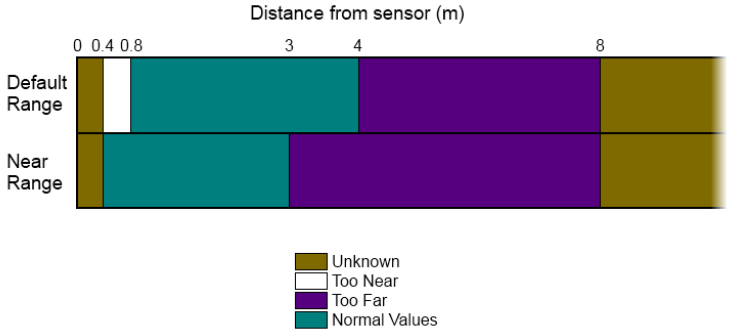

Figura 15: Rang de distancies estàndard del sensor de profunditat Kinect

Algunes aplicacions necessiten informació de profunditat més lluny o a prop dels límits de distància mostrats a la figura anterior, encara que la resolució o exactitud disminueixi. Per aconseguir això, es modifica el flux d'informació de dades de profunditat, de forma que es reportaran tots els píxels encara que es trobin al rang "Too Far" o "Too Near", mentre que els inclosos al rang "Unknown" seguiran mostrant un valor de 0.

9.2.2.2 Skeletal tracking

L'altra part imprescindible dins del SKD i que depèn completament de les dades proporcionades pel Depth Stream descrit anteriorment, són les llibreries destinades a realitzar el "Skeletal Tracking".

La funcionalitat Skeletal tracking (seguiment d'esquelet) és la funcionalitat estrella del sensor Kinect. Aquesta es basa en un algoritme que aconsegueix identificar les parts del cos de les persones que estan a dins del camp de visió del sensor. Mitjançant aquest algoritme es poden

obtenir punts que fan referència a les parts del cos d'una persona i fer un seguiment d'aquests, identificant gestos i postures. [11]

El SDK de Kinect permet obtenir els punts de les articulacions (Classe "*joint*") de l'esquelet i la seva posició d'una forma relativament fàcil. Per poder utilitzar el Skeleton tracking s'haurà d'afegir l'opció "UseSkeletalTracking" quan s'inicialitza el sensor, i es crearà l'event "SkeletonFrameReady" cada cop que s'obtingui una nova frame. Utilitzant un mètode que sigui cridat per aquest event serà possible capturar les dades de l'esquelet un cop s'obtinguin.

Els esquelets poden tenir 2 estats, "tracked" o "position only". Un esquelet en estat "tracked" conté informació de les 20 articulacions del cos de l'usuari que es trobin dins del camp de visió del sensor. Un esquelet en estat de "position only" només té informació de la posició de l'usuari, però no conté cap detall de les articulacions.

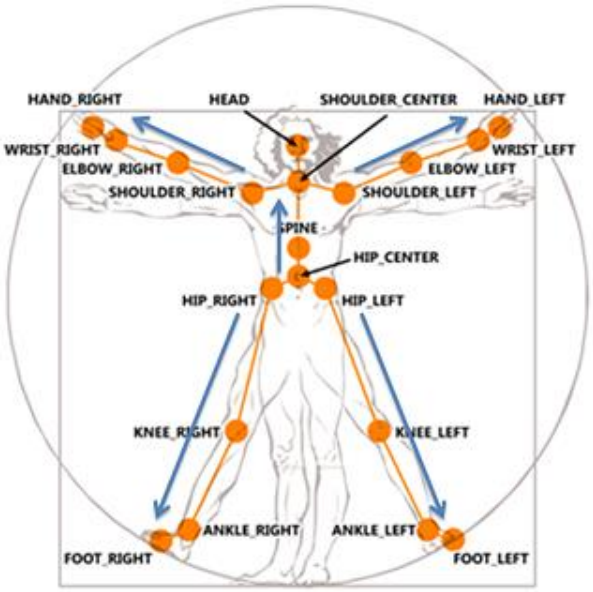

Figura 16: 20 Articulacions d'un esquelet en estat "tracked"

El SDK només dona l'opció d'accedir alhora a les dades de les articulacions de 2 esquelets dins del camp de visió, la resta de posicions del vector per on es reben els diferents esquelets capturats seran esquelets buits (NotTracked).

Per obtenir les posicions de les parts del cos cal utilitzar un enumerat anomenat *JointID* per tal d'obtenir les posicions d'un punt en concret. Per exemple, per obtenir la posició X de la mà dreta:

```
float HandRightX = skeleton.Joints[JointID.HandRight].Position.X;
```

Amb la propietat “Position” s’obté la posició respecte al sensor, seguit de l’eix de coordenades en la qual estem interessats. Amb les posicions de les diferents parts del cos es poden detectar postures i gestos utilitzant diferents tècniques.

9.2.3 Llibreria vector3 per C#

A causa de la naturalesa de l’aplicació a desenvolupar, és absolutament imprescindible calcular les posicions de les articulacions respecte a les altres. Tenint en compte que la classe “*Skeleton*” de les llibreries de Kinect SDK proporciona les dades d’aquestes en forma de punts espacials en 3 dimensions X,Y i Z, cal una forma de calcular els angles entre els diferents vectors formats per les principals parts articulades del cos humà.

Les llibreries “*Math*” de Windows per C#, només compten amb la classe “vector” que únicament permet treballar amb vectors de 2 dimensions, pel que ha sigut necessari utilitzar una llibreria addicional de lliure distribució per facilitar el treball en aquest aspecte.

La llibreria utilitzada ha sigut Vector3.cs, creada per Richard Potter. [9]

Està composta per un struct amb 3 variables privades (x,y,z) de tipus double. La llibreria està formada per constructors, definició d’operadors i altres mètodes per realitzar operacions matemàtiques entre vectors de 3 dimensions, entre els quals es troben: Interpolacions, distàncies, càlcul d’angles, rotacions, etc.

El mètode principal utilitzat d’aquesta llibreria és l’ “Angle” que permetrà calcular l’angle entre 2 vectors tridimensionals.

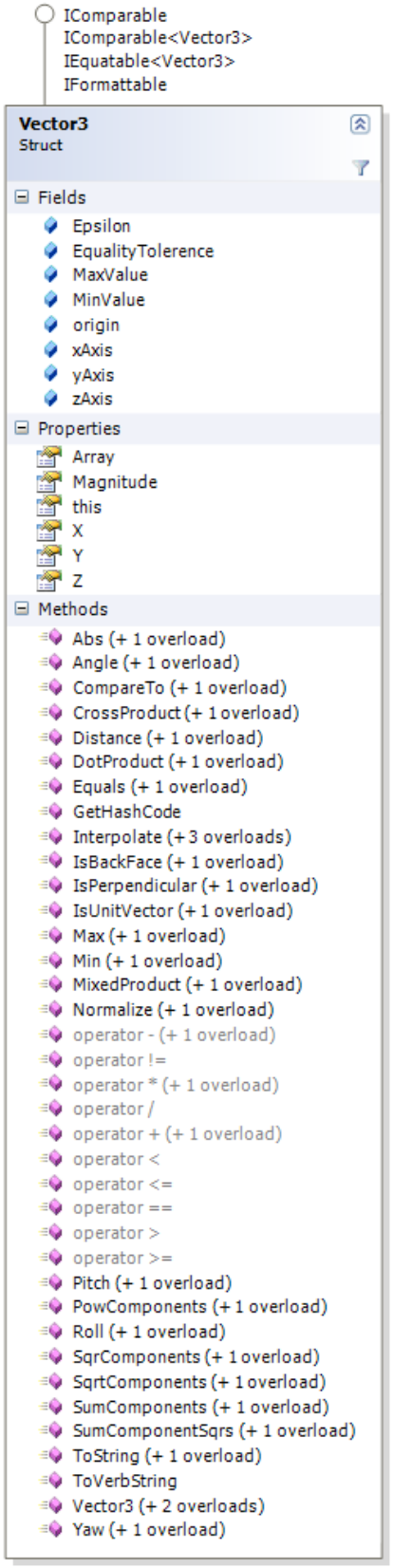

Figura 17: Descripció de la llibreria vector3

10 Anàlisi i disseny del sistema

L'aplicació final haurà de permetre visualitzar a través de la pantalla de l'ordinador la posició i postura de l'esquelet trobat pel sensor a partir del cos de l'usuari en temps real. També caldrà incloure un visualitzador del valor de tots els angles d'interès on es podrà visualitzar d'una forma precisa si el sensor està correctament calibrat.

S'haurà de poder seleccionar quin tipus de pràctica esportiva s'està realitzant i si l'usuari és dretà o esquerrà per l'activitat de boxa, tot de la forma més intuïtiva possible.

L'aplicació, si hem activat l'opció de "correcció postural esportiva", haurà de mostrar per pantalla les posicions de les articulacions l'usuari en verd si la postura és correcta o en vermell (les parts implicades) si s'està fent un moviment incorrecte perquè pugui ser corregit ràpidament.

També s'haurà d'indicar a l'usuari si alguna de les articulacions del seu esquelet queda fora del camp de visió del sensor perquè s'allunyi o ajusti aquest, així com un missatge d'error en cas de no trobar un dispositiu Kinect connectat a l'ordinador personal.

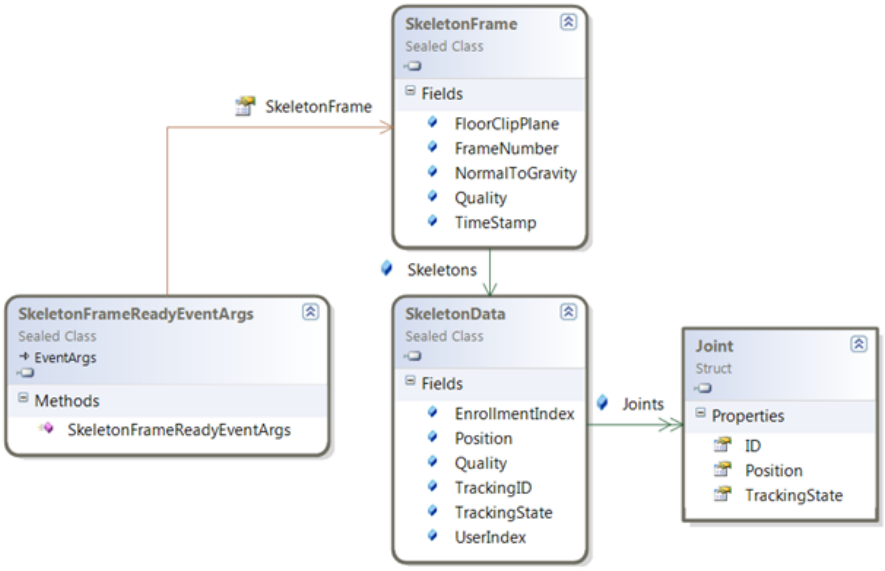

Figura 18: Diagrama de classes del SDK utilitzades

Per aconseguir això, s'utilitzaran les classes de la figura anterior. Un cop s'ha rebut un event indicant que s'ha capturat una nova frame pel sensor, de la "*SkeletonFrame*" s'obtindrà un vector que contindrà els diferents "*skeletons*" que han sigut captats pel sensor. Analitzant la "*SkeletonData*" de cada esquelet del vector, es podrà conèixer si el seu estat és "*tracked*" (conté dades dels "*joints"*) o pel contrari, només es coneix la seva posició.

Si l'estat de l'esquelet és "*tracked*" es podrà accedir al valor dels 20 "*Joints*" en cas que estiguin sent seguits, a través dels quals es poden obtenir les dades de posició per calcular el vector entre 2 d'ells.

Finalment, s'utilitzarà una operació matemàtica per obtenir els angles resultants entre la intersecció de dos d'ells. Amb el valor de cadascun d'aquests angles, tant de forma individual o respecte als altres, serà possible determinar si una posició concreta és vàlida o errònia.

10.1 Classes i mètodes principals utilitzats

En aquest apartat s'estudiaran quines classes, structs i altres estructures principals s'han utilitzat, fent una breu descripció de cadascuna, així com els mètodes principals utilitzats dins d'aquestes i també els propis creats per l'aplicació principal.

Com s'ha deixat entreveure a l'anàlisi anterior, a més de les classes principals, l'aplicació desenvolupada utilitza 5 classes (i 2 més que es mencionen, "Pen" i "Brush"), 5 structs i 4 "*enumeradors*" (jointType, JointTrackingState, TrackingState i TrackingID) en total.

- **Struct Point**
  - Aquest struct pertany a les llibreries de System.Windows.
  - Està format per 2 valors double X i Y.
  - És utilitzat a l'aplicació per tal de convertir un punt de profunditat obtingut del sensor "*DepthImagePoint*" a un punt d'aquest tipus per tal que el mètodes "*DrawLine*" i "*DrawEllipse*" de la classe "*DrawingContext*" puguin interpretar-lo correctament, i dibuixar a la imatge final que es mostra per pantalla les figures geomètriques adients.
- **Struct DepthImagePoint**
  - Aquest struct pertany a les llibreries de Microsoft.Kinect (SDK).
  - Està format per 4 ints, "*Depth"* (que indica el valor de la profunditat del píxel), *PlayerIndex* (número d'usuari a qui pertany el punt), i posicions X i Z.
  - Conté les dades tridimensionals d'un punt capturat pel sensor IR del dispositiu Kinect, i serà el pas intermedi per convertir un "*SkeletonPoint*" a un "*Point*" a través del següent métode:
    - *Public DepthImagePoint MapSkeletonPointToDepthPoint(SkeletonPoint, DepthImageFormat):* S'obtindrà la posició relativa d'un "*DepthImagePoint*" a través d'un "*SkeletonPoint*" i indicant la resolució en píxels de la imatge final.
- **Struct SkeletonPoint**
  - Aquest struct pertany a les llibreries de Microsoft.Kinect (SDK).
  - Està format per 3 valors float X, Y i Z.
  - És utilitzat a l'aplicació per conèixer les posicions a l'espai de les articulacions (*joints*) seguides i esquelets (*skeletons*) dels que només coneixem la seva posició però no estan seguits.
  - Tant el struct "*Joint"* com la classe "*Skeleton"* el contenen.
- **Struct Joint**
  - Aquest struct pertany a les llibreries de Microsoft.Kinect (SDK).
  - Està format per un int "*JointType*", que és una enumeració definida que indica el tipus d'articulació, per un "*JointTrackingState*" que també es un int obtingut de

una enumeració definida que ens indica si l'articulació està sent seguida o no i d'un "*SkeletonPoint*".
    - És utilitzat per conèixer totes les dades i característiques de les articulacions d'un esquelet, que pot contenir 20 tipus d'articulacions.
- **Classe JointCollection**
    - Aquesta classe pertany a les llibreries de Microsoft.Kinect (SDK).
    - Està format per enumerable que conté una taula de "*joints*" on s'emmagatzemaran per a cada esquelet.
    - És utilitzat a dins de la classe "*Skeleton*" per emmagatzemar els "*joints*" obtinguts de la frame capturada pel sensor, així com per obtenir les dades d'un "*joint*" en concret.
    - Conté el següent mètode rellevant:
        - *Public Joint this[JointType]:* Retorna o guarda un "*joint*" corresponent a la posició indicada pel enumerador JointType.
- **Classe Skeleton**
    - Aquest struct pertany a les llibreries de Microsoft.Kinect (SDK)
    - Està format per un "*JointCollection*" que conté les dades de tots els "joints", un "*SkeletonPoint*" que conté la posició de l'esquelet, un int "*TrackingID*" que indica el número d'usuari de l'esquelet i un int "*TrackingState*" que indica si l'esquelet està sent seguit o no.
    - És utilitzat a l'aplicació per guardar i accedir a les dades d'un esquelet.
    - Conté el següent métode rellevant:
        - *Public FrameEdges ClippedEdges():* Torna un FrameEdges, un int definit a una llista d'enumeradors, que ens indica si alguna articulació de l'esquelet es surt del camp de visió del sensor, sent l'enter per quina part ho fa.
- **Struct Vector3**
    - Aquest struct pertany a l'arxiu *Vector3.cs*
    - Está format per 3 variables doubles: X, Y i Z.
    - És utilitzat a l'aplicació per calcular el vector entre 2 "*joints*", així com per calcular l'angle format entre 2 vectors en particular.
    - Conté els següents mètodes rellevants:
        - *Public Vector3(double,double,double):* Constructor que crea un vector a partir dels seus 3 valors X,Y i Z.
        - *Public static double Angle(Vector 3, Vector3*): Retorna el valor de l'angle entre 2 vectors en radiants.
- **Classe SkeletonFrame**
    - Aquesta classe pertany a les llibreries de Microsoft.Kinect (SDK).
    - Aquesta classe està formada per un int "*FrameNumber*" que indica el número de frame capturada, "*SkeletonArrayLength*" que conté la llargada de la taula d'esquelets capturats, la taula "*SkeletonData*" que conté un taula d'esquelets així com altres variables de configuració no rellevants.

- S'obté a l'aplicació principal amb el métode "*OpenSkeletonFrame*()" cridat sobre la classe event "*SkeletonFrameReadyEventArgs*" cada cop que es rep una nova dada d'esquelet pel sensor.
- El mètode principal utilitzat dins d'aquesta classe és:
  - *Public void CopySkeletonDataTo(Skeleton[]):* Copia tots els esquelets a una taula.

- **Classe DrawingContext**
  - Aquesta classe pertany a les llibreries de System.Windows.Media.
  - Cal tenir en compte que la majoria de mètodes de la classe utilitzen les classes "*pen*" i "*brush*" (que es defineixen com a constants a l'inici de l'arxiu principal de l'aplicació *MainWindow.caml.cs*) per tal d'indicar les característiques de l'eina de dibuix que s'utilitza per realitzar
  - Aquesta classe abstracta de Windows serà utilitzada per tal de realitzar les imatges de sortida de l'aplicació gràcies als mètodes propis que conté que es descriuen a continuació:
    - *Public astract void DrawEllipse(Brush, Pen, Point, double, double):* Dibuixa una el·lipse al "*Point*" indicat, amb tamany X i Y indicats als "*doubles*" utilitzant les eines "*Brush*" i "*Pen*" entrades per paràmetre.
    - *Public astract void DrawLine(Pen, Point, Point):* Dibuixa una línia entre els 2 punts indicats amb l'eina "Pen" especificada.
    - *Public astract void DrawRectangle(Brush, Pen, Rect):* Dibuixa un rectangle (definit amb una altra classe) amb les eines entrades per paràmetre.
- **Classe DrawingImage**
  - Aquesta classe pertany a les llibreries de System.Windows
  - S'utilitza per convertir la classe "*DrawingContext*" en una "*DrawingImage*" per tal que l'aplicació pugui interpretar la imatge de sortida i mostrar-la per l'etiqueta "*image*" definida al codi XAML de l'aplicació.
  - Per aconseguir això s'utilitzarà el constructor de la classe pel que li passarem el "*DrawingContext*" per paràmetre.
- **Mètodes principals de "MainWindow.xaml.cs"**
  - *Private void WindowLoaded(sender, RoutedEventArgs)*
    - Mètode de tipus "*Event Handler*" que s'inicia al carregar l'aplicació. Activarà el sensor Kinect i el "skeletonStream"
  - *Private void WindowClosing(sender, CancelEventArgs)*
    - Mètode de tipus "*Event Handler*" que s'inicia al tencar l'aplicació. Desactivará el sensor Kinect.
  - *Private static void RectanglesClippedEdges(Skeleton, DrawingContext)*
    - Mètode encarregat de dibuixar rectangles a les cantonades de la imatge si alguna part de l'esquelet surt del camp de visió del sensor per aquesta.
  - *Private void SensorSkeletonFrameReady(sender, SkeletonFrameReadyEventArgs)*
    - Mètode de tipus "Event Handler" que s'inicia al rebre una nova "*Skeleton frame*" desde el sensor. Aquest s'encarrega d'obtenir els "*skeletons*" de la

frame rebuda i cridar el següent métode en cas de rebre un esquelet en estat "*tracked*". En cas de només conèixer la posició (estat "PositionOnly") dibuixarà una el·lipse blava a la posició d'aquest.

- *Private void DrawUnionsAndJoints(Skeleton, DrawingContext)*
  - Mètode principal encarregat de realitzar tots els càlculs necessaris per dibuixar l'esquelet de l'usuari per pantalla depenent de la sel·lecció de l'usuari. Per aconseguir això utilitza tots els mètodes que es descriuen a continuació.
- *Private Point SkeletonPoint(SkeletonPoint)*
  - Mètode que retorna un "*Point*" a partir d'un "*SkeletonPoint*" per tal de utilitzar-lo amb els mètodes de la classe "*DrawingContext*".
- *Private bool JointExists(Skeleton, JointType)*
  - Mètode que retorna "cert" si el tipus d'articulació existeix a l'esquelet. Necessari per fer qualsevol operació amb un joint per evitar qualsevol tipus d'error.
- *Private void DrawBone(Skeleton, DrawingContext, JointType, JointType)*
  - Mètode que dibuixa una línia verda gruixuda entre els 2 "*joints*" especificats de l'esquelet si tenen estat "*tracked*". Si només d'ells té aquest estat, dibuixa una línia simple, altrament no dibuixa cap línia.
- *Private void DrawBone2(Skeleton, DrawingContext, JointType, JointType)*
  - Mètode que dibuixa una línia vermella gruixuda entre els 2 "joints" especificats de l'esquelet si tenen estat "*tracked*". Si només d'ells té aquest estat, dibuixa una línia simple, altrament no dibuixa cap línia.
- *Private double angleValue3joints(Joint, Joint, Joint)*
  - Mètode que retorna el valor en angles de la intersecció del vector format pel segon i primer "*joint*", amb el tercer i segon "*joint*".

11 Implementació

11.1 Precisió i calibratge del sensor Kinect.

Abans de començar a establir els rangs de valors vàlids i invàlids d'angles per l'aplicació és necessari conèixer l'exactitud de les dades obtingudes pel sensor de IR del dispositiu Kinect. Els errors de mesura poden provenir de les següents fonts:

- El sensor.
- Condicions de l'entorn.
- Propietats de la superfície de l'objecte.

Ja que Kinect utilitza un sistema de triangulació per conèixer la distància a la qual es troba un objecte (emissor – objecte - receptor), els errors del sensor poden ser originats per un calibratge inadequat del dispositiu. Un calibratge inadequat o un error a l'estimació dels paràmetres pot comportar un error sistemàtic als punts de coordenades de cada punt. Aquests errors poden ser

eliminats per un calibratge adequat amb la utilització de programes de calibratge com *Kinect Calibration Toolbox* o *KinectStudio*, aplicació inclosa al *SDK*.

Els errors provocats per les condicions de l'entorn estan relacionats principalment per condicions de llum i la distància dels objectes. Les condicions de lluminositat influeixen directament amb l'aparició de disparitats, ja que sota una llum molt intensa, la sensibilitat del sensor disminueix. El fet de practicar les activitats esportives en un espai tancat fa que aquest error de mesura sigui menyspreable en aquest escenari, ja que es redueix molt el soroll provinent de les interferències lumíniques de l'aire lliure.

Encara que l'usuari estigui dins del rang d'operació vàlid del sensor, la distància a la qual es trobi d'aquest també influirà en l'exactitud de les mesures. L'error aleatori de mesura s'incrementa al augmentar la distància del sensor, fins a uns 4 cm a una distància de 5 metres d'aquest.

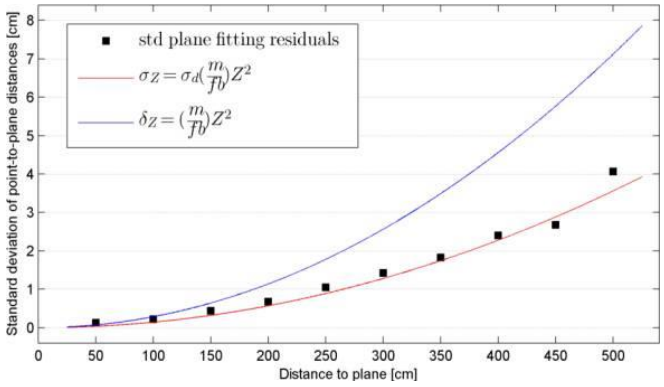

Figura 19: Error aleatori (vermell) i precisió de profunditat (blau) a diferents distàncies del sensor

Les propietats de la superfície de l'objecte també impacten en el mesurament del punts. Les superfícies brillants o molt polides provoquen errors de mesura al desviar el raig de llum infraroja. En aquest aspecte no és motiu de preocupació, ja que els punts d'interès tractats es troben sobre els cossos dels usuaris, on no trobarem aquest tipus de superfícies.

11.2 Càlcul de posicions del esquelet.

11.2.1 Angle entre dos vectors

Per tal de calcular els angles d'una part articulada del nostre cos a partir dels dos vectors que intersecten en aquesta, serà necessari realitzar una operació matemàtica per cadascun dels angles d'interès.

Primerament, es normalitzaran els vectors, a partir dels valors de cada component d'aquests dividits per la llargada total,

llargada = sqrt((ax * ax) + (ay * ay) + (az * az))
x = ax/|llargada|
y = ay/|llargada|
z = az/|llargada|

a continuació es realitzarà el producte escalar entre els 2 vectors,

$$\begin{vmatrix} i & j & k \\ a & b & c \\ d & e & f \end{vmatrix} = i \begin{vmatrix} b & c \\ e & f \end{vmatrix} - j \begin{vmatrix} a & c \\ d & f \end{vmatrix} + k \begin{vmatrix} a & b \\ d & e \end{vmatrix}$$

Figura 20: producte escalar entre 2 vectors 3D

i finalment es calcularà el arcosinus del resultat del producte escalar. La funció simplificada que s'utilitzarà a l'aplicació és la que es mostra a continuació, però multiplicant el resultat per (180/PI) per obtenir el valor en graus en comptes d'en radiants.

```
Math.Acos(Normalize(v1).DotProduct(Normalize(v2)));
```

11.2.2 Articulacions utilitzades

Per realitzar els càlculs sobre moviments correctes s'utilitzaran els angles de 12 parts articulades del cos humà, que es mostren contínuament per pantalla per fer un seguiment, especialment important durant la programació de l'aplicació, per tal de fer ajusts i correccions.

Els angles formats per parts articulades del cos que es tenen en compte per realitzar l'anàlisi postural, són els formats pels vectors que intersecten en un mateix punt (sent aquest punt un "*Joint*") que coincideix en una articulació d'interès per l'aplicació.

A continuació s'enumeren els diferents angles d'interès utilitzats de l'esquelet i com han sigut obtinguts:

- **Coll:** Vectors formats per les articulacions: Cap ←Centre d'espatlles i Columna baixa ←Centre d'espatlles.
- **Pelvis:** Vectors formats per les articulacions: Centre d'espatlles ←Columna baixa i Pelvis central ←Columna baixa.
- **Espatlla dreta:** "" Colze dret ←Espatlla dreta i Centre d'espatlles ←Espatlla dreta.
- **Espatlla esquerra:** ""Colze esquerre ←Espatlla esquerra i Centre d'espatlles ←Espatlla esquerre.
- **Colze dret:** "" Canell dret ←Colze dret i Espatlla dreta ←Colze dret.
- **Colze esquerre:** "" Canell esquerre ←Colze esquerre i Espatlla esquerra ←Colze esquerre.

- **Canell dret:** “” Mà dreta ←Canell dret i Colze dret ←Canell dret.
- **Canell esquerre:** “”Mà esquerra ←Canell esquerre i Colze esquerre ←Canell esquerre.
- **Cama dreta:** “” Genoll dret ←Pelvis dreta i Pelvis central ←Pelvis dreta.
- **Cama esquerra:** “” Genoll esquerre ←Pelvis esquerra i Pelvis central ←Pelvis esquerra.
- **Genoll dret:** “” Pelvis dreta ←Genoll dret i Turmell dret ←Genoll dret.
- **Genoll esquerre:** “” Pelvis esquerre ←Genoll esquerre i Turmell esquerre ←Genoll esquerre.

A causa de com posiciona la llibreria *skeleton.cs* les articulacions de l'esquelet a partir de les dades llegides pel sensor, en alguns angles dels descrits és més complex calcular la posició del cos de l'usuari, ja que aquestes articulacions estan situades a unes posicions que no s'ajusten a la realitat. Els angles afectats pel fet descrit són: Pelvis, Espatlla dreta, Espatlla esquerra, Cama dreta i Cama esquerra. La resta d'interseccions són més simples de calcular i ajustar, ja que un angle de 180º significarà que la articulació està totalment estirada.

El motiu pel qual s'ha ignorat el càlcul del "*joint*" corresponent al turmell ha sigut a causa del fet que Kinect té greus problemes a l'hora de calcular l'angle d'aquesta articulació quan el peu està en contacte amb el terra, donant un marge d'error massa elevat per considerar-lo, havent de descartar totes les correccions que el tenen en compte. Així que la llista final de les diferents correccions que s'han realitzat per cada activitat són les següents:

Córrer

- Genolls mai completament estirats.
- Cap recte mirant al davant.
- Esquena recta.
- Cama no massa avançada a la fase de suport.

Salt amb corda

- Genolls lleugerament flexionats.
- Cap recte sense mirar als peus.
- Espatlles estàtiques i properes al nostre tronc.

Boxa amb sac

- Cama avançada lleugerament flexionada (depenent de l'usuari).
- Colze mai completament estirat al colpejar.
- Turmell ben alineat amb el braç al llençar un cop de puny
- Es manté la guàrdia quan es llença un cop de puny.
- Si estem en guàrdia sense llençar cops, els dos genolls han d'estar lleugerament flexionats per augmentar la mobilitat.

12 Implantació i resultats

12.1 Desenvolupament de la interfície gràfica d'usuari

Utilitzant el llenguatge XAML s'ha desenvolupat una interfície gràfica d'usuari, el codi de la qual es pot trobar a l'arxiu *MainWindow.xaml* que permetrà visualitzar les dades i imatges necessàries, així com permetre seleccionar les diferents opcions descrites a l'apartat anterior.

El llenguatge declaratiu XAML ha permès configurar i personalitzar la interfície gràfica amb molta facilitat i d'una forma intuïtiva. A través dels valors dels elements introduïts en aquesta, identificables a partir del seu nom o etiqueta, el codi C# queda interconnectat amb aquest, enviant i rebent dades sempre que sigui necessari. A continuació es descriu la distribució i funció dels elements que es troben a la interfície gràfica.

Al centre de la pantalla principal apareixerà un quadre de 640x480 píxels, amb etiqueta "*Image*" per on es rebran els frames de sortida de l'aplicació, un cop realitzats tots els processos de càlcul i mostra de posicions.

A la part superior dreta es troba un selector de tipus "*slider*" amb etiqueta "*slValue*" on l'usuari podrà seleccionar l'activitat esportiva que vol practicar. Els valors que donarà aquest seran 0=Córrer, 1=Salt, 2=Boxa. Inicialment, per defecte, es trobarà seleccionada l'activitat de córrer. Llegint aquest valor, l'aplicació podrà conèixer quin tipus de correccions vol l'usuari. A sota d'aquest selector hi ha una "*checkbox*" amb l'etiqueta "*usuariEsquerra*", desactivada per defecte, que l'usuari haurà d'activar, en cas de ser esquerrà, per rebre una correcció adequada durant l'activitat de Boxa amb sac.

A la part dreta es troben els valors dels 12 angles descrits a l'apartat 11.2.2. Tots ells són "*textbox*" (caixes de text) amb nom "*MostrarAngle**", sent * el nom del joint del que es vol mostrar l'angle. A sota d'aquest apareix un "*slider*" amb etiqueta "*margeError*", amb valor 15 per defecte, que s'utilitzarà per determinar quin marge de graus de llibertat se li permet a l'usuari respecte a la posició ideal.

A la part inferior esquerra de la pantalla es pot veure una "*CheckBox*" amb l'etiqueta "*trackingOn*", per defecte activada, que permet activar o desactivar la correcció postural. Aquesta marcarà en vermell, si està activada, els errors a les articulacions depenent de l'esport seleccionat. A sota d'aquesta "*CheckBox*" apareix la línia de text "*Activa per corregir l´activitat seleccionada*" amb etiqueta "*StatusBarText*", que en el cas de no trobar un sensor Kinect connectat, mostrarà la línia de text "*Sensor Kinect no trobat!*".

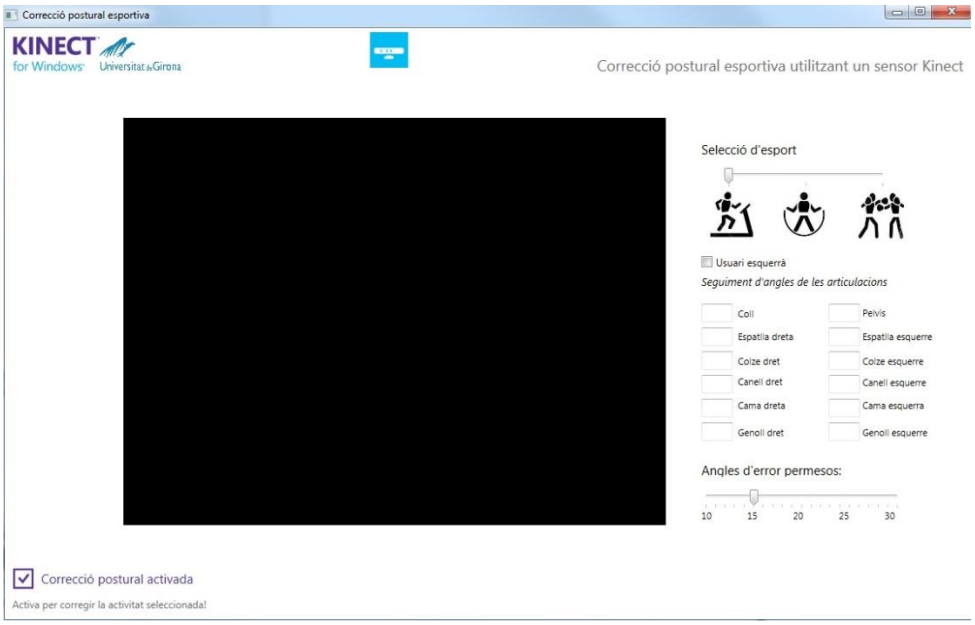

Figura 21: Interfície gràfica de l'aplicació

12.2 Captura de la imatge i posicions del esquelet

Per començar a capturar dades del sensor IR del dispositiu Kinect i transformar-les després en articulacions de l'esquelet que es pugin interpretar, s'haurà de seguir una sèrie de passes.

Com a nota important, ja que ha sigut una de les novetats de les quals ha sigut imprescindible documentar-se i entendre per realitzar aquest projecte, cal explicar que són els "*Event Handlers*" que s'utilitzen a C#, i en especial en aquest codi.
Un *Event Handler* és un tipus de mètode que serà cridat automàticament en resposta a un event concret, com podrien ser per exemple una nova dada introduïda per l' usuari o nova informació rebuda pel sensor.

Un cop carregada la pantalla de la interfície gràfica, aquesta enviarà un event al codi i s'iniciarà el mètode "*WindowLoaded(event)*". Dins de les diferents funcions d'aquest mètode, la primera serà detectar si hi ha un o diferents sensors Kinect connectats i en cas de trobar-ne més d'un utilitzarem el primer trobat. En cas contrari, la textbox amb etiqueta"statusBarText" a la interfície gràfica, mostrarà que no s'ha trobat cap sensor Kinect. En cas de tancar la finestra de l'aplicació, en qualsevol moment de l'execució, s'envia un event al mètode "*WindowClosed*(event)" que s'encarregarà de desconnectar el sensor Kinect en cas d'existir.

Seguidament, s'activarà el *SkeletonStream* de Kinect amb el mètode "*SkeletonStream.Enable()"* que permetrà començar a rebre dades d'esquelets a través del sensor.

A continuació es passarà a cridar el "*event Handler*" "*SensorSkeletonFrameReady(event)*"que serà cridat exteriorment sempre que hi hagi una nova *SkeletonFrame* llesta des del sensor. Per realitzar això s'utilitzarà l'operador "+=" que és equivalent a subscriure un event.

A dins del mètode "*SensorSkeletonFrameReady(event)"* es crea una taula de *Skeletons* on es guardaran les dades dels esquelets rebuts al nou frame, que podran ser fins a 2 en estat "*Tracked*". S'ha mantingut aquesta funcionalitat activada pensant en noves aplicacions futures, encara que l'aplicació està pensada per corregir a una sola persona.

Finalment es comprovarà si existeix un esquelet a la primera posició de la taula (usuari) i si està sent seguit amb la variable "*TrackingState*". En cas afirmatiu, ja tenim una frame d'un esquelet amb els valors de la posició dels 20 "*joints*" (si tots ells es troben també en estat "*Tracked*").

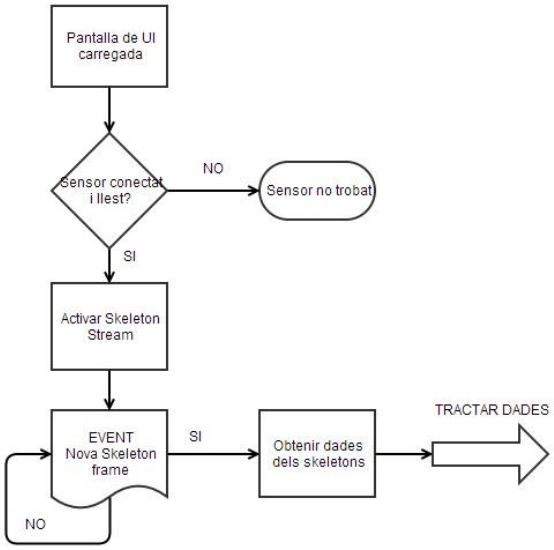

Figura 22: Diagrama de flux de la inicialització de la captura de dades

12.3 Tractament de dades i mostra de posicions correctes per cada sector del cos

Aquest apartat és el més dens de tot el codi, ja que inclou totes les passes que hi han des de la rebuda de les dades en forma de "*Skeletons*", passant pel càlcul de postures per cada part del cos tractada en cada activitat esportiva, fins a la mostra per pantalla amb les correccions fetes.

El pas inicial és crear un "*drawingContext*" de 640x480 píxels, classe obtinguda de les llibreries de sistema "*System.Windows.Media*", que permetrà realitzar dibuixos i utilitzar mètodes de dibuix geomètric a sobre d'aquest. Aquest "*drawingContext*" s'enllaçarà a l'objecte amb etiqueta "*image*" a la UI de l'aplicació i és on l'usuari podrà visualitzar totes les dades gràficament.

Un cop obtingudes les dades d'un esquelet a partir d'una frame capturada pel sensor (de la forma explicada a l'apartat anterior) es dibuixarà un rectangle negre de 640x480 que serà el fons de la imatge on es mostrarà el resultat gràfic final.
En comptes d'un fons negre es podria haver utilitzat la imatge capturada per la càmera RGB, activant el "*ColorStreamRGB*" del sensor Kinect, i actualitzant el fons cada cop que rebéssim una frame nova. Aquesta opció s'ha descartat perquè ha comportat una disminució de FPS significativa i en escenes molt lluminoses dificulta la visualització del gràfic de l'esquelet mostrat per pantalla.

Seguidament es cridarà el mètode "*RectanglesClippedEdges(skeleton, drawingContext*)" que rebrà com a paràmetres el "*drawingContext*" descrit i el "Skeleton" que s'estigui tractant. Aquest mètode s'encarregarà de dibuixar un rectangle vermell al llarg dels costats superior, inferior, dret i/o esquerre del "*drawingContext*" en cas que algun "*joint*" de l'esquelet sobresurti del camp de visió del sensor, fent impossible determinar la localització d'aquest. Per aconseguir això s'utilitzarà a dins del mètode descrit, el mètode "*ClippedEdges.HasFlag*(*position*)" de la classe "*skeleton*" que indicarà si esta succeint o no per cadascuna de les direccions.

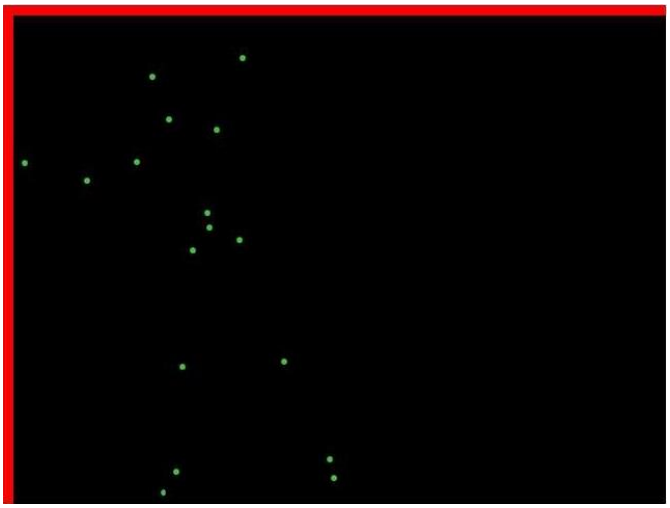

Figura 23: Usuari amb articulacions fora del camp de visió del sensor per la part superior i dreta

A continuació, l'aplicació examinarà si l'esquelet està sent seguit o pel contrari, només tenim la posició d'aquest, utilitzant la variable *trackingState* de la classe "*skeleton*". En cas de només tenir la posició d'aquest, com pot succeir quan el sensor està començant a captar les primeres dades d'un nou usuari, es dibuixarà una circumferència blava al "*drawingContext*" a la posició on el sensor Kinect ha calculat que està centrat l'esquelet. En cas de que l'esquelet estigui sent seguit, i per tant tingui l'estat "*tracked*", s'invocarà el mètode "*DrawUnionsAndJoints(drawingContext, skeleton)*" que rebrà per paràmetre el "*Skeleton*" tractat i el "*drawingContext*" on dibuixarem l'esquelet.

12.3.1 Dibuixant unions i articulacions

El primer pas que es realitzarà al mètode "*DrawUnionsAndJoints(drawingContext, skeleton*)" serà mostrar numèricament per la UI, a cadascun dels quadres amb etiquetes "*mostrarAngleX*", l'angle corresponent a les 12 articulacions d'interès descrites a l'apartat 11.2.2. Per fer això, primerament

es comprovarà que els "*joints*" implicats en el càlcul de l'angle de l'articulació existeixen amb el mètode "*JointExists(skeleton, jointType)*" i seguidament es cridarà al mètode "*angleValue3joints(joint, joint, joint)*" que rebrà els 3 joints implicats en l'articulació en format "Joint extrem A – Joint Central – Joint extrem B" i tornarà un double amb el valor de l'angle en graus.

Figura 24: Angles mostrats per la UI durant l'execució

Un altre pas que realitzarà el mètode serà el de dibuixar les 20 articulacions (totes les possibles) que existeixin del "*Skeleton*". El primer que es farà serà comprovar l'estat "*TrackingState*" de cada "*Joint*" tractat, per conèixer si es "*Inferred*" (interferit per una part del cos).

Quan passi això, es dibuixarà un punt groc al "*drawingContext*" amb el mètode "*DrawEllipse*" de la classe *drawingContext*, pel qual es passarà per paràmetre la posició del punt, el color i la mida d'aquest, per donar entendre que la posició de l'articulació tractada no és del tot fiable, mentre que si no està interferida, s'utilitzarà el mateix mètode per dibuixar un punt verd clar.

Per tal de saber si l'usuari vol activar les correccions o no, es comprovarà la checkbox "*trackedOn*" de la UI i a continuació es comprovarà el valor del slider amb etiqueta "*slValue*" per saber quin tipus d'activitat esportiva ha seleccionat l'usuari.

Un cop realitzat aquest pas es passarà a executar la part encarregada de dibuixar les unions entre les articulacions, específica per cada activitat, que avaluarà diferents factors que s'expliquen en el següent apartat.

Quan existeixin 2 "*joints*" d'un esquelet que formen una unió, tal com el existent entre el genoll dret i el turmell dret, per exemple, es cridarà el mètode "*DrawBone(drawingContext, skeleton, jointType, jointType*)" que rebrà per paràmetre el "*drawingContext*", el "*skeleton*" i els 2 tipus de "*joints*" en cas de que es vulgui dibuixar una línia verda al "*drawingContext*" entre aquestes 2 articulacions (pels casos on es realitza un moviment correcte); i es cridarà al mètode *"DrawBone2(drawingContext, skeleton, jointType, jointType)*" en cas de voler dibuixar una línia vermella per la resta de casos.

Els 2 mètodes comprovaran, abans de dibuixar la línia, si algun dels 2 "*joints*" que formen part de la unió està interferit (el mateix procediment que es segueix quan es dibuixa cada articulació). En cas d'estar-ho una d'elles, es dibuixarà una línia simple (de color blanc i estret), i en cas d'estar-ho les dues no dibuixaran cap línia, ja que al desconèixer la posició real de l'articulació, no es pot córrer el risc d'afirmar si el moviment realitzat és correcte o no.

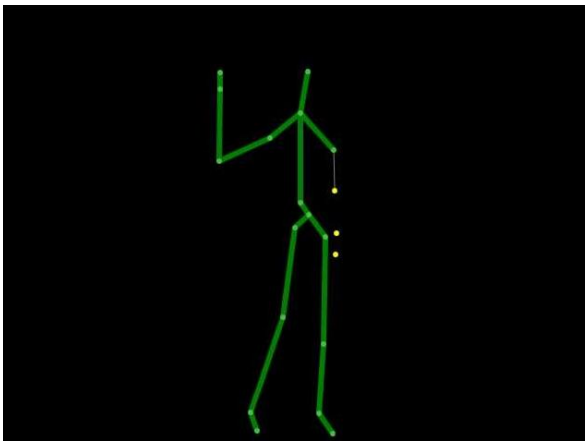

Figura 25: Usuari amb el colze, canell i mà esquerres interferits

12.3.2 Correccions a les activitats esportives

Per cada activitat esportiva caldrà que l'usuari estigui en una posició específica respecte al sensor per captar les dades correctament. Ja que el sac de boxa i el panell de la cinta de córrer es troben sempre davant de l'usuari i interfereixen amb la visió del sensor Kinect, caldrà que aquest últim estigui en una posició lleugerament cap al costat l'usuari, tenint en compte que com més s'acosti a la perpendicular d'aquest, més articulacions quedaran superposades. Durant l'activitat de salt en corda, com que no existeixen aquests impediments, el sensor haurà d'estar enfront de l'usuari.

Cal tenir en compte que a l'hora de valorar si una postura és correcta o no, existirà un marge d'error que s'afegirà respecte a la posició correcta ideal de l'angle de l'articulació en tots els casos. Aquest marge d'error, que per defecte és 15 graus, pot ser modificat utilitzant el slider de la UI descrit anteriorment.

Detectarem errors a l'activitat de córrer sobre cinta si:

- Qualsevol dels 2 colzes està massa estirat o flexionant, partint com a la posició ideal un angle de 90º. Quedaran marcades les unions Canell↔Colze↔Espatlla.
- Coll està massa flexionat, partint com a posició ideal un angle de 180º, l'equivalent a una mirada fixa al front. Quedarà marcada la unió Cap↔Centre d'espatlles.
- Esquena està massa doblegada, partint com a posició ideal 135º (ja que la "*joint*" de la pelvis no esta alineada en 180º amb la de la columna). Queda marcada la unió Centre d'espatlles↔Columna.
- Qualsevol de les 2 cames està massa avançada respecte al cos, a partir de 125º. Queda marcada la unió Turmell↔Genoll↔Pelvis.

Detectarem errors a l'activitat de salt amb corda si:

- Coll està massa flexionat, partint com a posició ideal un angle de 180º, l'equivalent a una mirada fixa al front. Quedarà marcada la unió Cap↔Centre d'espatlles.
- Qualsevol de les 2 espatlles es mouen amunt o abaix, partint com a posició ideal un angle de 145º (les 3 "*joints*" corresponents a les espatlles no estan alineades). Queda marcada la unió Colze↔Espatlla↔Centre d'espatlles.
- Qualsevol dels 2 genolls està completament estirat, considerant estirat un angle de 185º (5º de més, ja que es vol ser més estricte després d'aplicar els angles de marge d'error). Queda marcada la unió Turmell↔Genoll↔Pelvis.

Detectarem errors a l'activitat de boxa amb sac si:

- Quan els 2 colzes estan doblegats (guàrdia), no es tenen els genolls doblegats. Entenent com a genolls estirats un angle de 180º. Queda marcada la unió Pelvis↔Genoll↔Turmell.
- Quan els 2 colzes estan estirats més de 90º, s'entén que l'usuari ha deixat de cobrir el seu cap. Queda marcada la unió Cap↔Centre d'espatlles.
- Quan s'ha llençat un cop de puny (només un dels 2 colzes té un angle inferior a 90º):
  A) Si el colze del braç estirat arriba a 180º, s'entén que l'usuari ha sobre-estirat el braç. Queda marcada la unió Espatlla↔Colze↔Canell.
  B) Si el canell del braç del cop no està estirat del tot, proper a 180º. Queda marcada la unió Ma↔Canell.
  C) Si la cama avançada (esquerra per dretans i dreta per esquerrans) està estirada, considerant un angle de 180º al genoll. Queda marcada la unió Turmell↔Genoll↔Pelvis.

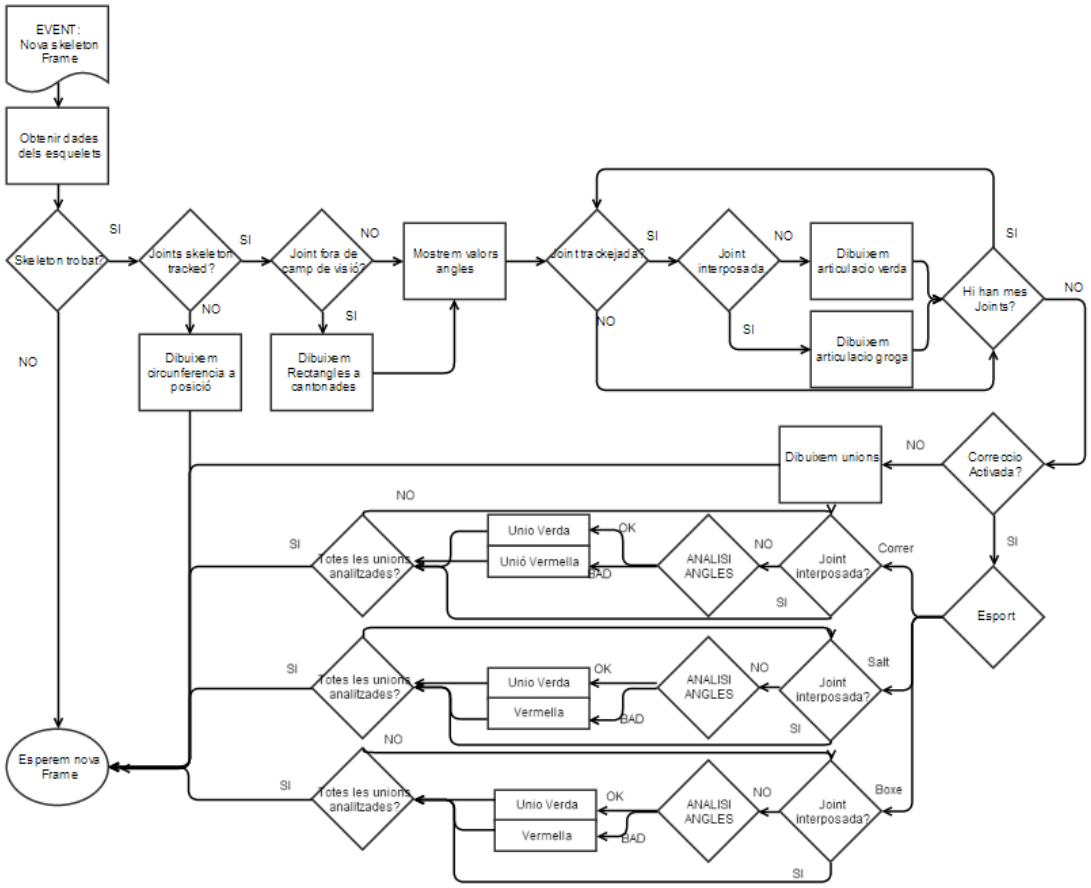

Figura 26: Diagrama de flux simplificat del tractament i mostra de dades

12.4 Resultats

En aquest apartat es mostraran els resultats finals de l'aplicació en diferents exemples per cada cas d'ús.

Cal tenir en compte que les proves i ajustaments necessaris, perquè les correccions fetes s'acostessin als objectius buscats, s'han realitzat a mesura que es desenvolupava cada part del codi. Hi ha hagut un total de 14 versions diferents de l'aplicació fins a la que es presenta en aquest PFC.

Cada cas d'ús s'ha testejat practicant l'activitat esportiva a casa, excepte l'activitat de córrer, que s'ha testejat un dia a dins d'un gimnàs.

Com a punt final cal destacar que la posició del sensor Kinect respecte a l'usuari és una variable important respecte a la qualitat de les dades rebudes respecte a les posicions reals del cos, així com els moviments massa bruscs.

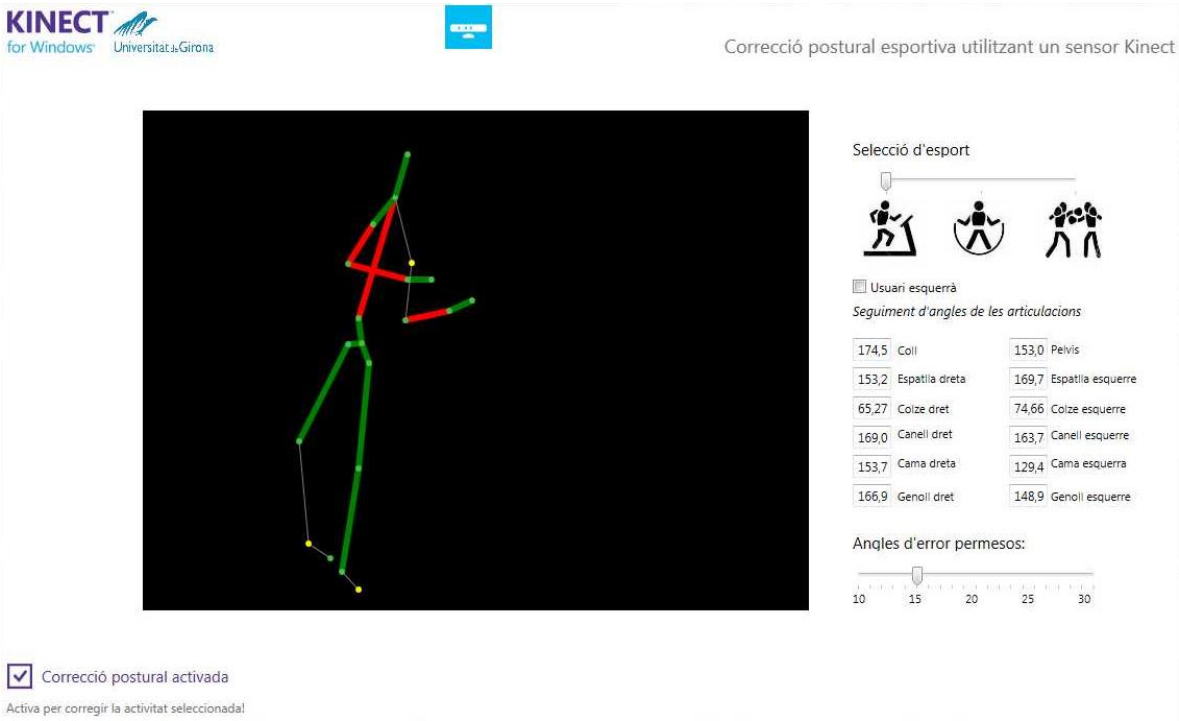

Figura 27: Usuari corrents amb la esquena i els colzes massa doblegats

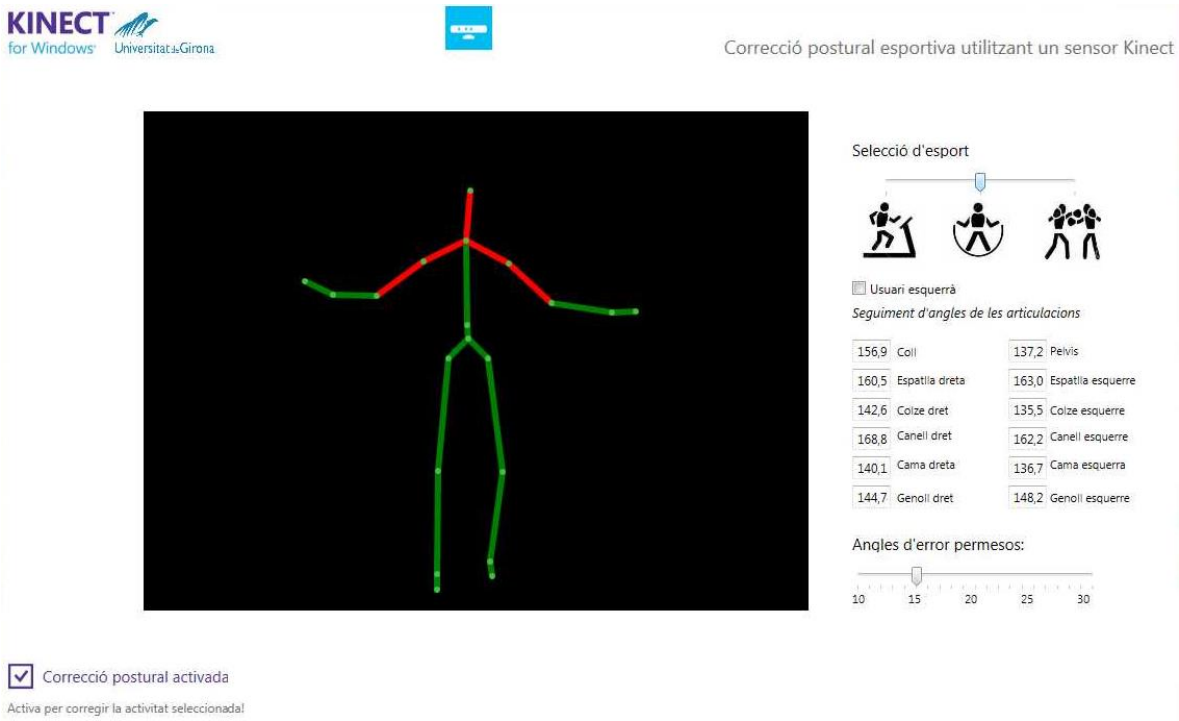

Figura 28: Usuari saltant la corda movent les espatlles i mirant-se els peus

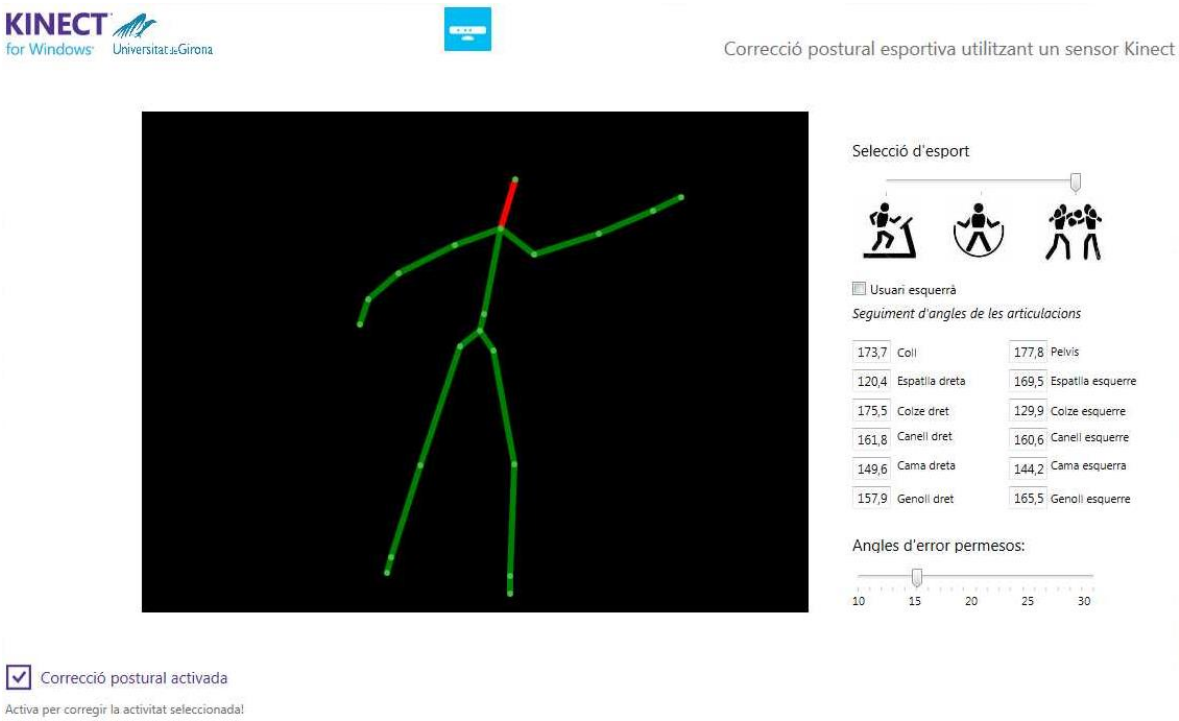

Figura 29: Usuari practicant boxa sense protegir-se el cap

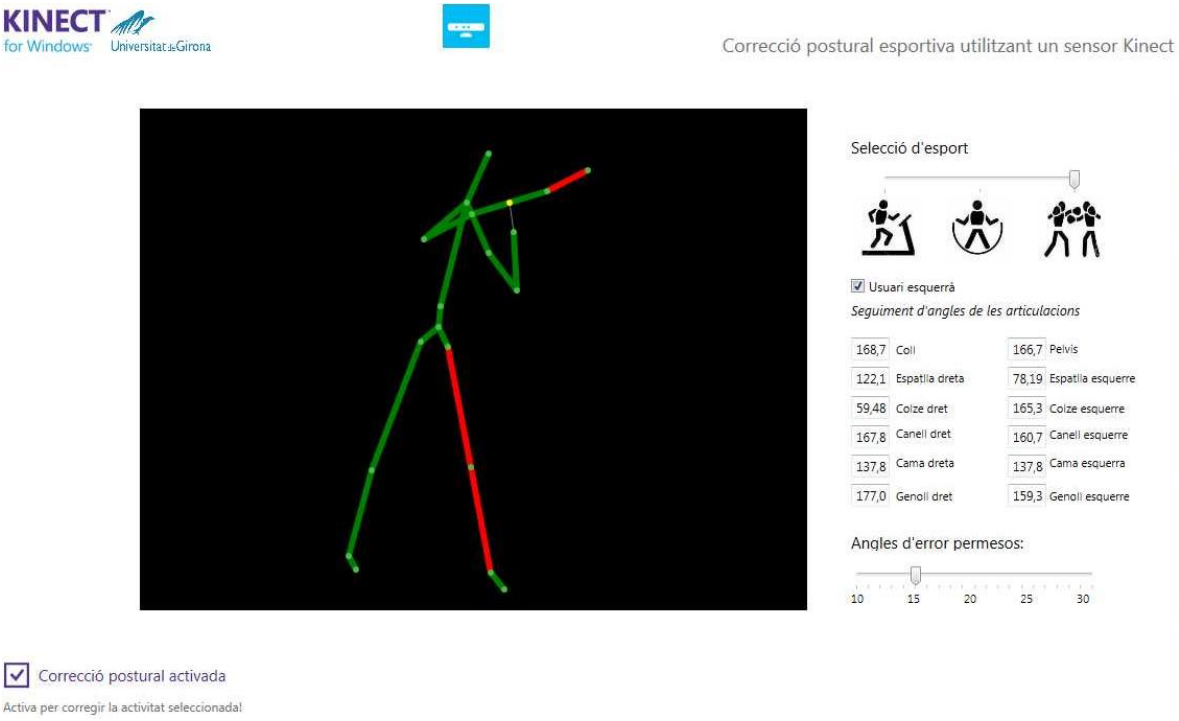

Figura 30: Usuari esquerrà picant amb la cama avançada recta i el canell doblegat

13 Conclusions

L'objectiu principal del projecte era el reconeixement i correcció de postures incorrectes durant la pràctica d'una serie d'activitats esportives. Es pot concloure que s'ha aconseguit complir aquest objectiu de manera satisfactòria, encara que algunes de les correccions per a extremitats concretes (en el cas de l'articulació del turmell, a causa dels motius descrits a l'apartat 11.2.2) no s'han pogut tenir en compte per les característiques del dispositiu.

Totes les llibreries aportades pel *SDK de Kinect*, encara que han requerit una gran quantitat de temps per la seva comprensió, han simplificat molt el treball.

Ha sigut un exercici molt interessant que m'ha permès aprendre nous conceptes respecte al funcionament de dispositius de visió per computador, així com les diferències del llenguatge C# respecte a altres més familiars, així com aprendre el llenguatge XAML, completament nou per mi, amb gran utilitat dins d'entorns Windows.

13.1 Problemes trobats i solucions aportades

- A causa del fet que el motor elèctric que permet el moviment vertical del sensor Kinect no ha sigut dissenyat per un ús exhaustiu, té una limitació de moviments que impedeix realitzar més d'un moviment cada 15 segons. Per aquest motiu s'ha hagut de descartar el mètode "*AdjustVerticalAngle(Skeleton)"* (es troba comentat al codi) que permetia autoajustar el sensor quan l'usuari es trobava massa elevat o baix respecte al camp de visió del sensor. Per substituir aquest es va utilitzar el mètode "*RectanglesClippedEdges"* ja explicat.
- El sensor Kinect té molts de problemes a l'hora d'intentar determinar l'angle d'una articulació que està molt propera a una superfície, ja que es provoquen molts errors de mesura insalvables. Per culpa d'això, l'angle de l'articulació corresponent als turmells no ha pogut ser utilitzat per fer correccions de postures que inicialment, durant la fase de documentació sobre activitats esportives, es tenien planejades.
- El sensor té més dificultat en analitzar una escena a on l'usuari es troba lateralment i no frontalment, ja que moltes de les articulacions queden superposades per altres parts del cos. Per aquest motiu s'ha decidit no tenir en compte les unions entre articulacions quan almenys una d'elles està interposat per una altra part del cos.
- En augmentar la distància del sensor, la inexactitud dels angles mostrats augmenta, motiu pel qual es va incorporar el slider que permet afegir més angles de llibertat al que l'aplicació considera com un moviment correcte.

14 Treball futur

En aquest apartat s'enumeren una sèrie de possibles millores que es podrien realitzar a l'aplicació en un futur:

- Activació dels selectors d'activitat esportiva i tracking per veu, utilitzant el software de reconeixement de veu del SDK.
- Incorporació d'un segon sensor Kinect per augmentar la precisió de les dades.
- Modificació de la imatge mostrada per la interfície gràfica, passant aquesta a ser un model 3D del cos seguit.
- Noves opcions pel selector d'esports, que puguin incloure esports amb 2 persones i que corregeixin errors posturals basant-se en dades de l'esquelet propi i dades de l'altre usuari recíprocament.

15 Bibliografia

1. Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision. Prentice Hall, 1998.*
2. E. Roy Davies (2005). *Machine Vision: Theory, Algorithms, Practicalities. Morgan Kaufmann. .*
3. http://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx
4. http://www.asus.com/Multimedia/Xtion_PRO_LIVE/
5. http://www.i3du.gr/pdf/primesense.pdf
6. B. Curless. *From range scans to 3d models. ACM SIGGRAPH Computer Graphics, 33(4):38–41, 1999.*
7. http://en.wikipedia.org/wiki/3D_scanner
8. Khoshelham, K. and Oude Elberink, S.J. . *In: Sensors : journal on the science and technology of sensors and biosensors : open access, 12 (2012)2 pp. 1437-1454.*
9. http://www.codeproject.com/Articles/17425/A-Vector-Type-for-C
10. http://msdn.microsoft.com/es-es/library/52f3sw5c(v=vs.90).aspx
11. http://msdn.microsoft.com/en-us/library/hh855347.aspx
12. *Z. Lewis. The Plain Dealer (May 10, 2011)*
13. www.running.es
14. Arnheim, D., (1994) *Medicina deportiva, Fisioterapia y entrenamiento atlético, Mosby/Doyma Libros, División de Times Mirror de España, S.A., España.*
15. http://www.jumprm.com/
16. http://www.expertboxing.com/
17. K. Khoshelham and S. Oude Elberink. *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications.*

16 Annexos

16.1 MainWindow.xaml.cs

```
namespace Microsoft.Kinect.CorreccioEsportiva
{
    using System;
    using System.IO;
    using System.Windows;
    using System.Windows.Media;
    using Microsoft.Kinect;


    /// Interaction logic for MainWindow.xaml
    public partial class MainWindow : Window
    {
        /// Ample del dibuix de sortida (quadre negre on es dibuixarà esquelet,
s'amplia cap a la dreta)
        private const float RenderWidth = 640.0f;

        /// Altura del dibuix de sortida
        private const float RenderHeight = 480.0f;

        /// Gruix de les unions
        private const double JointThickness = 3;

        /// Gruix de la elipsis central
        private const double BodyCenterThickness = 10;

        /// Gruix dels rectangles per indicar que esquelet surt del quadre.
        private const double ClipBoundsThickness = 10;

        /// Color de la circunferencia/eclipse quan esquelet no estigui traquejat
        private readonly Brush centerPointBrush = Brushes.Blue;

        /// Color del pinzell utilitzat per dibuixar Joints que estan siguent
trackejades (verd clar)
        private readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68));

        /// Color del pinzell utilitzat per dibuixar joints que estan siguent
superposades
        private readonly Brush inferredJointBrush = Brushes.Yellow;


        /// Color i gruix utilitzat per osos que están siguent trackejats
        private readonly Pen trackedBonePen = new Pen(Brushes.Green, 6);

        /// Color i gruix utilitzat per osos que están siguent interferits per
altres parts del esquelet
        private readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);

        /// Color i gruix utilitzat per osos que están siguent trackejats
        private readonly Pen angleBonePen = new Pen(Brushes.Red, 6);


        /// Sensor de Kinect ACTIU
        private KinectSensor sensor;
```

```
        /// Grup de dibuix per sortida de renderitzat d'esquelet
        private DrawingGroup drawingGroup;
        /// Imatge dibuixada que mostrarem
        private DrawingImage imageSource;


        /// Creem una nova instancia de MainWindow class. (Pantalla de la aplicacio
principal)
        public MainWindow()
        {
            InitializeComponent();
        }


        //Mètode d'autocalibració del motor del sensor Kinect DESCARTAT degut a la
limitació del motor trobada.
        /*private void AdjustVerticalAngle(Skeleton skeleton)
        {
                //Si esquelet es surt per la part baixa i no per la alta, i angle
del sensor vertical es mes gran que -27 graus
            if ((skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom)) &&
(!skeleton.ClippedEdges.HasFlag(FrameEdges.Top)) && (this.sensor.ElevationAngle > (-
27)))
            {
                int verticalAngle=this.sensor.ElevationAngle;
                verticalAngle=verticalAngle-2;
                if (verticalAngle >= (-27))
                {
                    this.sensor.ElevationAngle = verticalAngle;
                }
                else{this.sensor.ElevationAngle = -27;}
            }

            //Si esquelet es surt per la part alta i no per la baixa, i angle del
sensor vertical es mes petit que 27 graus
            if ((!skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom)) &&
(skeleton.ClippedEdges.HasFlag(FrameEdges.Top)) && (this.sensor.ElevationAngle <
(27)))
            {
                int verticalAngle = this.sensor.ElevationAngle;
                verticalAngle=verticalAngle+2;
                if (verticalAngle <= (27))
                {
                    this.sensor.ElevationAngle = verticalAngle;
                }
                else { this.sensor.ElevationAngle = 27; }
            }
        }*/


        /// Dibuixa rectangles indicadors a les cantonades del display per mostrar
si part de l'esquelet no apareix.
        private static void RectanglesClippedEdges(Skeleton skeleton, DrawingContext
drawingContext)
        {
            /// Si esquelet surt per part baixa
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom))
            {
```

```
                ///Dibuixem rectangle amb (Color,null,Nou rectangle=(posicio
X0,Y0,Xmax,yMax,Gruix))
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, RenderHeight - ClipBoundsThickness, RenderWidth,
ClipBoundsThickness));
            }
            /// Si esquelet surt per part alta
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Top))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, RenderWidth, ClipBoundsThickness));
            }
            /// Si esquelet surt per part esquerra
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Left))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, ClipBoundsThickness, RenderHeight));
            }
            /// Si esquelet surt per part dreta
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Right))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(RenderWidth - ClipBoundsThickness, 0,
ClipBoundsThickness, RenderHeight));
            }
        }


        // Tasques de inicialització del sensor
        // Pantalla principal de la UI carregada -> Event
        private void WindowLoaded(object sender, RoutedEventArgs e)
        {
            // Creem el nou grup de dibuix que utilitzarem per dibuixar.
            this.drawingGroup = new DrawingGroup();
            // Creem una font de imatge que utilitzarem en el nostre control de
imatge
            this.imageSource = new DrawingImage(this.drawingGroup);
            // Mostrem el dibuix utilitzant el nostre control de imatge
            Image.Source = this.imageSource;


            // Buscarem tots els posible sensors conectats y utilitzarem el primer
conectat.
            // Requereix que el sensor estigui conectat abans de iniciar la
aplicació.
            foreach (var potentialSensor in KinectSensor.KinectSensors)
            {
                if (potentialSensor.Status == KinectStatus.Connected)
                {
                    this.sensor = potentialSensor;
```

```
                    break;
                }
            }

            // Si existeix un sensor conectat llavors...
            if (null != this.sensor)
            {
                // Activarem el "Skeleton Stream" a aquest sensor per començar a
rebre frames del esquelet
                this.sensor.SkeletonStream.Enable();
                // Afegim un event handler que será cridat sempre que hi hagi noves
dades de tipues skeleton frames (actualitzarà esquelet rebut)
                // "The use of the += operator in this context is referred to as
subscribing to an event."
                this.sensor.SkeletonFrameReady += this.SensorSkeletonFrameReady;

                // Inicialitzem el sensor!
                try
                {
                    this.sensor.Start();
                }
                catch (IOException)
                {
                    this.sensor = null;
                }
            }
            // Si no es troba sensor, missatge No Kinect Ready
            if (null == this.sensor)
            {
                this.statusBarText.Text =Properties.Resources.NoKinectReady;
            }
        }


        /// Tasques de finalització del sensor
        // Al tencar la finestra principal de la aplicacio -> Event
        private void WindowClosing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            //Si sensor no Null, finalitzarem.
            if (null != this.sensor)
            {
                this.sensor.Stop();
            }
        }

        /// Event Handler pel sensor Kinect cridat a métode "void WinwowLoaded",
quan s'ha rebut una frame nova
        private void SensorSkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            //Creem taula "skeletons", creem nou skeleton a posició 0 de la taula
            Skeleton[] skeletons = new Skeleton[0];

            //Si tenim una nova SkeletonFrame a SkeletonFrameReadyEventArgs e, el
guardarem al final de la taula "skeletons"
            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
```

```
                if (skeletonFrame != null)
                {
                    skeletons = new Skeleton[skeletonFrame.SkeletonArrayLength];
//Modifiquem llargada de la taula "skeletons" amb el nombre de esquelets rebuts
                    skeletonFrame.CopySkeletonDataTo(skeletons);  //Copiem les dades
obtingudes de la SkeletonFrame obtinguda de "e" al nou skeleton guardat a la taula
"skeletons"
                }
            }


            using (DrawingContext dc = this.drawingGroup.Open())
            {
                // Dibuixem un rectangle transparent (fondo negre) per seleccionar
el espai de renderitzat
                dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0,
RenderWidth, RenderHeight));

                //Si hi ha algún esquelet guardat a la taula "skeletons"
                if (skeletons.Length != 0)
                {
                    //Per cada Skeleton a la taula "skeletons"...
                    foreach (Skeleton skel in skeletons)
                    {
                        RectanglesClippedEdges(skel, dc); //Cridem método per
dibuixar rectangles a cantonades on esquelet surti de pantalla

                        //Si el skeleton esta siguent Trackejat, cridarem método
DrawBonesAndJoints.
                        if (skel.TrackingState == SkeletonTrackingState.Tracked)
                        {
                            this.DrawUnionsAndJoints(skel, dc);

                        }
                        //Si només tenim la posició dibuixarem una
elipse/circunferencia
                        else if (skel.TrackingState ==
SkeletonTrackingState.PositionOnly)
                        {
                            dc.DrawEllipse(
                            this.centerPointBrush,
                            null,
                            this.SkeletonPointToScreen(skel.Position),
                            BodyCenterThickness,
                            BodyCenterThickness);
                        }
                    }
                }

                // Per prevenir dibuixar fora de la área de renderitzat, dibuixem
nou rectangle 640x480 començant desde les posicions X=0 i Y=0
                this.drawingGroup.ClipGeometry = new RectangleGeometry(new Rect(0.0,
0.0, RenderWidth, RenderHeight));
            }
        }


        /// Métode que dibuixara articulacions i joints de tot l' esquelet
```

```
        /// skeleton ->esquelet a dibuixar
        /// drawingContext ->Drawing context on dibuixarem
        private void DrawUnionsAndJoints(Skeleton skeleton, DrawingContext
drawingContext)
        {

            //Mostrar angles d'articulacions per pantalla
            //Es necessari comprobar sempre si el joint existeix, ja que si no es
aixi i val 0, la operacio provocarà un error a la aplicacio.

            // Braç esquerre
            //Mostrar angle Colze esquerre
            if ((JointExist(skeleton, JointType.ShoulderLeft)) &&
(JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.WristLeft)))
            { MostrarAngleColzeE.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft])); }
            //Mostrar angle Canell esquerre
            if ((JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.WristLeft)) && (JointExist(skeleton, JointType.HandLeft)))
            { MostrarAngleCanellE.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ElbowLeft],
skeleton.Joints[JointType.WristLeft], skeleton.Joints[JointType.HandLeft])); }
            //Mostrar angle Espatlla esquerra
            if ((JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.ShoulderLeft)) && (JointExist(skeleton, JointType.ShoulderCenter)))
           { MostrarAngleEspatllaE.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ElbowLeft],
skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ShoulderCenter])); }

            // Braç dret
            //Mostrar angle Colze dret
            if ((JointExist(skeleton, JointType.ShoulderRight)) &&
(JointExist(skeleton, JointType.ElbowRight)) && (JointExist(skeleton,
JointType.WristRight)))
            { MostrarAngleColzeD.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight])); }
            //Mostrar angle Canell dret
            if ((JointExist(skeleton, JointType.ElbowRight)) &&
(JointExist(skeleton, JointType.WristRight)) && (JointExist(skeleton,
JointType.HandRight)))
            { MostrarAngleCanellD.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ElbowRight],
skeleton.Joints[JointType.WristRight], skeleton.Joints[JointType.HandRight])); }
            //Mostrar angle Espatlla dreta
            if ((JointExist(skeleton, JointType.ElbowRight)) &&
(JointExist(skeleton, JointType.ShoulderRight)) && (JointExist(skeleton,
JointType.ShoulderRight)))
            { MostrarAngleEspatllaD.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ElbowRight],
skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ShoulderCenter])); }

            //Cama esquerra
            //Mostrar angle Cama esquerre
```

```
            if ((JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.HipLeft)) && (JointExist(skeleton, JointType.HipCenter)))
            { MostrarAngleCamaE.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.KneeLeft],
skeleton.Joints[JointType.HipLeft], skeleton.Joints[JointType.HipCenter]))); }
            //Mostrar angle Genoll esquerre
            if ((JointExist(skeleton, JointType.AnkleLeft)) && (JointExist(skeleton,
JointType.KneeLeft)) && (JointExist(skeleton, JointType.HipLeft)))
            { MostrarAngleGenollE.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.AnkleLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.HipLeft]))); }

            //Cama dreta
            //Mostrar angle Cama dreta
            if ((JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.HipRight)) && (JointExist(skeleton, JointType.HipCenter)))
            { MostrarAngleCamaD.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.KneeRight],
skeleton.Joints[JointType.HipRight], skeleton.Joints[JointType.HipCenter]))); }
            //Mostrar angle Genoll dret
            if ((JointExist(skeleton, JointType.AnkleRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.HipRight)))
            { MostrarAngleGenollD.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.AnkleRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.HipRight]))); }

            //Mostrar angle Coll
            if ((JointExist(skeleton, JointType.Head)) && (JointExist(skeleton,
JointType.ShoulderCenter)) && (JointExist(skeleton, JointType.Spine)))
            { MostrarAngleColl.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.ShoulderCenter], skeleton.Joints[JointType.Spine]))); }
            //Mostrar angle Pelvis
            if ((JointExist(skeleton, JointType.ShoulderCenter)) &&
(JointExist(skeleton, JointType.Spine)) && (JointExist(skeleton,
JointType.HipCenter)))
            { MostrarAnglePelvis.Text = string.Format("{0:0.00}",
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.Spine], skeleton.Joints[JointType.HipCenter]))); }



            //Cridem métode DrawBone per cada unió d'articulacions depenent de la
seleccio del usuari "trackingOn i slValue
            if (trackingOn.IsChecked == true){
            if (slValue.Value == 0){ //Si volem realitzar correccions posturals i la
activitat seleccionada es correr
                //Dibuixar parts Esquena, espatlles, Pelvis
                // Neck: Si coll esta massa doblegat o massa estirat, avisem d'error
                if ((JointExist(skeleton, JointType.Head)) && (JointExist(skeleton,
JointType.ShoulderCenter)) && (JointExist(skeleton, JointType.Spine)))
                {
                    if (((180 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.ShoulderCenter], skeleton.Joints[JointType.Spine]))) ||
((180 + margeError.Value) < (angleValue3joints(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.ShoulderCenter], skeleton.Joints[JointType.Spine])))
```

```
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                    }else{
                        this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                    }}
                //Spine: Si esquena està massa doblegada, avisem d'error
                if ((JointExist(skeleton, JointType.ShoulderCenter)) &&
(JointExist(skeleton, JointType.Spine)) && (JointExist(skeleton,
JointType.HipCenter)))
                {
                    if ((135 + margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.Spine], skeleton.Joints[JointType.HipCenter])))
                    {
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.Spine);
                    }else{
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.Spine);
                    }}

                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
                this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

                //Dibuixar parts Braç esquerre
                // Left Arm: Si colze esta massa doblegat o massa estirat, avisem
d'error
                if ((JointExist(skeleton, JointType.ShoulderLeft)) &&
(JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.WristLeft))){
                    if (((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft]))) ||
((90 + margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft]))))
                    {
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone2(skeleton, drawingContext,
JointType.ElbowLeft, JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
                    }else{
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
```

```
                    this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
                }}


                //Dibuixar parts Braç dret
                //Right Arm: Si colze esta massa doblegat o massa estirat, avisem
d'error
                if ((JointExist(skeleton, JointType.ShoulderRight)) &&
(JointExist(skeleton, JointType.ElbowRight)) && (JointExist(skeleton,
JointType.WristRight)))
                {
                    if (((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight])) ||
((90 + margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight]))))
                    {
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone2(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                    }else{
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                    }}

                //Cama esquerra
                //LeftLeg: Si cama esquerra esta massa adelatada respecte el cos
                if ((JointExist(skeleton, JointType.HipCenter)) &&
(JointExist(skeleton, JointType.HipLeft)) && (JointExist(skeleton,
JointType.KneeLeft)))
                {
                    if ((125 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.HipCenter],
skeleton.Joints[JointType.HipLeft], skeleton.Joints[JointType.KneeLeft])))
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
                        this.DrawBone2(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
                        this.DrawBone2(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                    }else{
                        this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);
                    }}
```

```
                // Cama dreta
                // Right Leg: Si cama dreta esta massa adelantada respecte el cos
                if ((JointExist(skeleton, JointType.HipCenter)) &&
(JointExist(skeleton, JointType.HipRight)) && (JointExist(skeleton,
JointType.KneeRight)))
                {
                    if ((125 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.HipCenter],
skeleton.Joints[JointType.HipRight], skeleton.Joints[JointType.KneeRight])))
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
                        this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                        this.DrawBone2(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                    }else{
                        this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
                        this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                    }}



            }else if (slValue.Value == 1){  //Si volem realitzar correccions
posturals i la activitat seleccionada es salt
                //Dibuixar parts Esquena, espatlles, Pelvis
                // Neck: Si coll esta massa doblegat , avisem d'error
                if ((JointExist(skeleton, JointType.Head)) && (JointExist(skeleton,
JointType.ShoulderCenter)) && (JointExist(skeleton, JointType.Spine)))
                {
                    if (((180 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.ShoulderCenter], skeleton.Joints[JointType.Spine]))) ||
((180 + margeError.Value) < (angleValue3joints(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.ShoulderCenter], skeleton.Joints[JointType.Spine])))
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                    }else{
                        this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                    }}



                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
                this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
```

```
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

                //Dibuixar parts Braç esquerre
                //Espatlles: estàtiques a la posició correcta
                if ((JointExist(skeleton, JointType.ShoulderLeft)) &&
(JointExist(skeleton, JointType.ShoulderCenter)) && (JointExist(skeleton,
JointType.ElbowLeft)))
                {
                    if (((145 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.ShoulderLeft], skeleton.Joints[JointType.ElbowLeft]))) ||
((145 + margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.ShoulderLeft], skeleton.Joints[JointType.ElbowLeft])))
                    {
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderLeft);
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
                    }else{
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderLeft);
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
                    }}



                //Dibuixar parts Braç dret
                //Espatlles: estàtiques a la posició correcta
                if ((JointExist(skeleton, JointType.ShoulderRight)) &&
(JointExist(skeleton, JointType.ShoulderCenter)) && (JointExist(skeleton,
JointType.ElbowRight)))
                {
                    if (((145 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.ShoulderRight], skeleton.Joints[JointType.ElbowRight])))
|| ((145 + margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderCenter],
skeleton.Joints[JointType.ShoulderRight], skeleton.Joints[JointType.ElbowRight])))
                    {
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderRight);
                        this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
```

```
                    }
                    else
                    {
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderCenter, JointType.ShoulderRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                    }
                }

                // Cama esquerra
                // Genoll esquerre: Si el genoll esta completamnt estirat avisar
d'error
                if ((JointExist(skeleton, JointType.HipLeft)) &&
(JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.AnkleLeft)))
                {
                    if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.AnkleLeft])))
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
                        this.DrawBone2(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);
                    }else{
                        this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);
                    }}

                // Cama dreta
                // Genoll dret: Si el genoll esta completamnt estirat avisar d'error
                if ((JointExist(skeleton, JointType.HipRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.AnkleRight)))
                {
                    if ((185 - (30-margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.AnkleRight])))
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
                        this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                    }
                    else
```

```
                    {
                        this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
                        this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                    }
                }

            }else if (slValue.Value == 2){  //Si volem realitzar correccions
posturals i la activitat seleccionada es boxa
                //Flux:: Si 2 colzes flexionats -> (estem en guardia) -> 2 genolls
doblegats
                //       Altrament si 1 colze flexionat i l'altre no -> (estem
llençant un cop) -> "Cama adelantada ha d'estar doblegada, l'altra normal" + "Colze
estirat ha de tenir un angle <180 i l'altre ha d estar doblegat mantenint la
guardia"

                //Dibuixar parts Esquena, espatlles, Pelvis
                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
                this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
                this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
                this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);


                if ((JointExist(skeleton, JointType.ShoulderLeft)) &&
(JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.WristLeft)) && (JointExist(skeleton, JointType.ShoulderRight)) &&
(JointExist(skeleton, JointType.ElbowRight)) && (JointExist(skeleton,
JointType.WristRight)))
                {
                    //Si 2 colzes flexionats (Guardia)
                    if (((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight]))) &&
((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft]))))
                    {
                        this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                        // Cama esquerra
                        // Genoll esquerre: Si el genoll esta completamnt estirat
avisar d'error
                        if ((JointExist(skeleton, JointType.HipLeft)) &&
(JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.AnkleLeft)))
                        {
```

```
                            if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.AnkleLeft])))
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                this.DrawBone2(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                            }
                            else
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                            }
                        }

                        // Cama dreta
                        // Genoll dret: Si el genoll esta completamnt estirat avisar
d'error
                        if ((JointExist(skeleton, JointType.HipRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.AnkleRight)))
                        {
                            if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.AnkleRight])))
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                            }
                            else
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                            }
                        }

                        //Dibuixar parts Braç esquerre
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
```

```
                        //Dibuixar parts Braç dret
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);



                    } //Si nomes colze dret esta doblegat (cop de puny esquerre)
                    else if ((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight]))
                    {
                        this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                        //Si es dretà, cama esquerra sempre llegerament flexionada
                        if (usuariEsquerra.IsChecked == false)
                        {
                            // Cama esquerra
                            // Genoll esquerre: Si el genoll esta completamnt
estirat avisar d'error
                            if ((JointExist(skeleton, JointType.HipLeft)) &&
(JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.AnkleLeft)))
                            {
                                if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.AnkleLeft])))
                                {
                                    this.DrawBone2(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                    this.DrawBone2(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                                }
                                else
                                {
                                    this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                                }
                            }

                            //Cama dreta normal
                            this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                            this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                            this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
```

```
                        }
                        else
                        { //Si usuari es esquerrà, cama dreta lleugerament
flexionada
                            // Cama dreta
                            // Genoll dret: Si el genoll esta completamnt estirat
avisar d'error
                            if ((JointExist(skeleton, JointType.HipRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.AnkleRight)))
                            {
                                if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.AnkleRight])))
                                {
                                    this.DrawBone2(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                    this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                                }
                                else
                                {
                                    this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                                }
                            }
                            // Cama esquerra normal
                            this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                        }

                        // Braç esquerre
                        // Colze esquerre: Si el colze esta completamnt estirat
avisar d'error
                        if ((JointExist(skeleton, JointType.ShoulderLeft)) &&
(JointExist(skeleton, JointType.ElbowLeft)) && (JointExist(skeleton,
JointType.WristLeft)))
                        {
                            if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft])))
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                                this.DrawBone2(skeleton, drawingContext,
JointType.ElbowLeft, JointType.WristLeft);
                            }
                            else
```

```
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.ElbowLeft, JointType.WristLeft);
                            }
                        }
                        //Canell esquerre: Si canell no esta completament estirat
avisar d'error
                        if (JointExist(skeleton, JointType.HandLeft)){
                            if (((180 - margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ElbowLeft],
skeleton.Joints[JointType.WristLeft], skeleton.Joints[JointType.HandLeft])) &&
((180 + margeError.Value) > (angleValue3joints(skeleton.Joints[JointType.ElbowLeft],
skeleton.Joints[JointType.WristLeft], skeleton.Joints[JointType.HandLeft]))))
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.WristLeft, JointType.HandLeft);
                            }
                            else
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.WristLeft, JointType.HandLeft);
                            }

                        }


                        //Dibuixar parts Braç dret
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);


                    } //Si nomes colze esquerre esta doblegat (cop de puny dret)
                    else if ((90 - margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ElbowLeft], skeleton.Joints[JointType.WristLeft])))
                    {
                        this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
                        //Si es dretà, cama esquerra sempre llegerament flexionada
                        if (usuariEsquerra.IsChecked == false)
                        {
                            // Cama esquerra
                            // Genoll esquerre: Si el genoll esta completamnt
estirat avisar d'error
                            if ((JointExist(skeleton, JointType.HipLeft)) &&
(JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.AnkleLeft)))
                            {
                                if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.AnkleLeft])))
                                {
```

```
                                this.DrawBone2(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                this.DrawBone2(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                            }
                            else
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                            }
                        }

                        //Cama dreta normal
                        this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);

                    }
                    else
                    { //Si usuari es esquerrà, cama dreta lleugerament
flexionada
                        // Cama dreta
                        // Genoll dret: Si el genoll esta completamnt estirat
avisar d'error
                        if ((JointExist(skeleton, JointType.HipRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.AnkleRight)))
                        {
                            if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.AnkleRight])))
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                            }
                            else
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                            }
                        }
```

```
                            // Cama esquerra normal
                            this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                        }

                        // Braç dret
                        // Colze dret: Si el colze esta completamnt estirat avisar
d'error
                        if ((JointExist(skeleton, JointType.ShoulderRight)) &&
(JointExist(skeleton, JointType.ElbowRight)) && (JointExist(skeleton,
JointType.WristRight)))
                        {
                            if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.ElbowRight], skeleton.Joints[JointType.WristRight])))
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                                this.DrawBone2(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                            }
                            else
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                                this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                            }
                        }
                        //Canell dret: Si canell no esta completament estirat avisar
d'error
                        if (JointExist(skeleton, JointType.HandRight))
                        {
                            if (((180 - margeError.Value) <
(angleValue3joints(skeleton.Joints[JointType.ElbowRight],
skeleton.Joints[JointType.WristRight], skeleton.Joints[JointType.HandRight])) &&
((180 + margeError.Value) >
(angleValue3joints(skeleton.Joints[JointType.ElbowRight],
skeleton.Joints[JointType.WristRight], skeleton.Joints[JointType.HandRight]))))
                            {
                                this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                            }
                            else
                            {
                                this.DrawBone2(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                            }

                        }

                        //Dibuixar parts Braç esquerre
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
```

```
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);


                    } //2 braços estirats, hem perdut la guardia, som vulnerables a
nockeig.
                    else
                    {
                        this.DrawBone2(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter); //
                        //Si es dretà, cama esquerra sempre llegerament flexionada
                        if (usuariEsquerra.IsChecked == false)
                        {
                            // Cama esquerra
                            // Genoll esquerre: Si el genoll esta completamnt
estirat avisar d'error
                            if ((JointExist(skeleton, JointType.HipLeft)) &&
(JointExist(skeleton, JointType.KneeLeft)) && (JointExist(skeleton,
JointType.AnkleLeft)))
                            {
                                if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.KneeLeft], skeleton.Joints[JointType.AnkleLeft])))
                                {
                                    this.DrawBone2(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                    this.DrawBone2(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                                }
                                else
                                {
                                    this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                                }
                            }

                            //Cama dreta normal
                            this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                            this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                            this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);

                        }
                        else
                        { //Si usuari es esquerrà, cama dreta lleugerament
flexionada
                            // Cama dreta
```

```
                            // Genoll dret: Si el genoll esta completamnt estirat
avisar d'error
                            if ((JointExist(skeleton, JointType.HipRight)) &&
(JointExist(skeleton, JointType.KneeRight)) && (JointExist(skeleton,
JointType.AnkleRight)))
                            {
                                if ((185 - (30 - margeError.Value)) <
(angleValue3joints(skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeRight], skeleton.Joints[JointType.AnkleRight])))
                                {
                                    this.DrawBone2(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                    this.DrawBone2(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                                }
                                else
                                {
                                    this.DrawBone(skeleton, drawingContext,
JointType.HipRight, JointType.KneeRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.KneeRight, JointType.AnkleRight);
                                    this.DrawBone(skeleton, drawingContext,
JointType.AnkleRight, JointType.FootRight);
                                }
                            }
                            // Cama esquerra normal
                            this.DrawBone(skeleton, drawingContext,
JointType.HipLeft, JointType.KneeLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.KneeLeft, JointType.AnkleLeft);
                            this.DrawBone(skeleton, drawingContext,
JointType.AnkleLeft, JointType.FootLeft);
                        }

                        //Usuari ja ha comès el error més greu al llençar el cop, no
evaluarem la sobre-elongacio del cop de puny
                        //Dibuixar parts Braç esquerre
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderLeft, JointType.ElbowLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
                        this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);
                        //Dibuixar parts Braç dret
                        this.DrawBone(skeleton, drawingContext,
JointType.ShoulderRight, JointType.ElbowRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.ElbowRight, JointType.WristRight);
                        this.DrawBone(skeleton, drawingContext,
JointType.WristRight, JointType.HandRight);
                    }
                }


            }
```

```
            }else{                    //Si no volem realitzar correcions posturals,
simplement dibuixem unions
            //Dibuixar parts Esquena, espatlles, Pelvis

            this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
            this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

            //Dibuixar parts Braç esquerre
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderLeft,
JointType.ElbowLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
            this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);

            //Dibuixar parts Braç dret
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderRight,
JointType.ElbowRight);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowRight,
JointType.WristRight);
            this.DrawBone(skeleton, drawingContext, JointType.WristRight,
JointType.HandRight);

            // Cama esquerra
            this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
            this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);

            // Cama dreta
            this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
            this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleRight,
JointType.FootRight);

            }


            // Articulacions
            foreach (Joint joint in skeleton.Joints)
            {
                //Creem nou pincell per dibuixar articulacions
```

```
                Brush drawBrush = null;

                //Si articulacions están trackejadees, pincell d'un color (verd clar
per defecte)
                if (joint.TrackingState == JointTrackingState.Tracked)
                {
                    drawBrush = this.trackedJointBrush;
                }
                //Si articulacions están superposades, `pincell d'un altre color
(groc per defecte)
                else if (joint.TrackingState == JointTrackingState.Inferred)
                {
                    drawBrush = this.inferredJointBrush;
                }
                //Si articulacions no estan trackejades ni interposades, dibuixarem
circunferencia del color indicat pel pincell drawBrush
                if (drawBrush != null)
                {
                    drawingContext.DrawEllipse(drawBrush, null,
this.SkeletonPointToScreen(joint.Position), JointThickness, JointThickness);
                }
            }
}




        /// Mapeja un esquelet per que aparegui a dins del nostre espai de
renderizat i el converteix a un punt. (Quan no el tenim trackejat, només la posició)
        /// <param name="skelpoint">point to map</param>
        /// <returns>mapped point</returns>
        private Point SkeletonPointToScreen(SkeletonPoint skelpoint)
        {
            // Convert point to depth space.
            // We are not using depth directly, but we do want the points in our
640x480 output resolution.
            DepthImagePoint depthPoint =
this.sensor.CoordinateMapper.MapSkeletonPointToDepthPoint(skelpoint,
DepthImageFormat.Resolution640x480Fps30);
            return new Point(depthPoint.X, depthPoint.Y);
        }


        private bool JointExist(Skeleton skeleton, JointType jointType0)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            if (joint0.TrackingState == JointTrackingState.NotTracked)
            {
                return false;
            }
            else
            {
                return true;
            }}


        /// <summary>
```

```
        /// Dibuixa una linia entre 2 articulacions
        /// </summary>
        /// <param name="skeleton">Skeleton del que dibuixarem linies</param>
        /// <param name="drawingContext">Drawing context on les dibuixarem</param>
        /// <param name="jointType0">Joint on iniciarem la linia</param>
        /// <param name="jointType1">Joint on acabarem la linia</param>
        private void DrawBone(Skeleton skeleton, DrawingContext drawingContext,
JointType jointType0, JointType jointType1)
        {
            //Creem 2 noves joints a partir de les entrades per paràmetre
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];

            // Si no podem trobar alguna de les articulacions, sortim.
            if (joint0.TrackingState == JointTrackingState.NotTracked ||
                joint1.TrackingState == JointTrackingState.NotTracked)
            {
                return;
            }

            // No dibuixarem la unió si les dues articulacions están superposades
per altres parts del esquelet
            if (joint0.TrackingState == JointTrackingState.Inferred &&
                joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }

            // Asumirem que tots els ossos dibuixats estan interferits, excepte si
les 2 articulacions estan trackejades
            Pen drawPen = this.inferredBonePen;
            // Si les 2 articulacions estan trackejades... verd
            if (joint0.TrackingState == JointTrackingState.Tracked &&
joint1.TrackingState == JointTrackingState.Tracked)
            {
                drawPen = this.trackedBonePen;
            }

            drawingContext.DrawLine(drawPen,
this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }


        /// Dibuixa una linia entre 2 articulacions en posició incorrecta (vermell)
        /// <param name="skeleton">Skeleton del que dibuixarem linies</param>
        /// <param name="drawingContext">Drawing context on les dibuixarem</param>
        /// <param name="jointType0">Joint on iniciarem la linia</param>
        /// <param name="jointType1">Joint on acabarem la linia</param>
        private void DrawBone2(Skeleton skeleton, DrawingContext drawingContext,
JointType jointType0, JointType jointType1)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];

            // Si no podem trobar alguna de les articulacions, sortim.
            if (joint0.TrackingState == JointTrackingState.NotTracked ||
                joint1.TrackingState == JointTrackingState.NotTracked)
```

```
            {
                return;
            }

            // No dibuixarem la unió si les dues articulacions están superposades
per altres parts del esquelet
            if (joint0.TrackingState == JointTrackingState.Inferred &&
                joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }

            // Asumirem que tots els ossos dibuixats estan interferits, excepte si
les 2 articulacions estan trackejades
            Pen drawPen = this.inferredBonePen;
            // Si les 2 articulacions estan trackejades... vermell
            if (joint0.TrackingState == JointTrackingState.Tracked &&
joint1.TrackingState == JointTrackingState.Tracked)
            {
                drawPen = this.angleBonePen;
            }

            drawingContext.DrawLine(drawPen,
this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }

        /// Retorna el valor de l'angle a partir de 3 joints.
        /// jointB sempre será la articulació central d'una extremitat o part del
cos (la que volem mesurar).
        /// Els vectors surten de la articulacio per tal de que els calculs siguin
corretes
        private double angleValue3joints(Joint jointA, Joint jointB, Joint jointC)
        {
            Vector3 a1 = new
Vector3(jointA.Position.X,jointA.Position.Y,jointA.Position.Z);
            Vector3 a2 = new Vector3(jointB.Position.X, jointB.Position.Y,
jointB.Position.Z);
            Vector3 a3 = new Vector3(jointC.Position.X, jointC.Position.Y,
jointC.Position.Z);

            Vector3 b1 = a1 - a2;
            Vector3 b2 = a3 - a2;

            return (b1.Angle(b2))*(180/(Math.PI));
        }

    }
}
```

16.2 MainWindow.xaml

```
<Window x:Class="Microsoft.Kinect.CorreccioEsportiva.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Correcció postural esportiva" Height="735" Width="1156"
Loaded="WindowLoaded" Closing="WindowClosing">

    <Window.Resources>
        <SolidColorBrush x:Key="MediumGreyBrush" Color="#ff6e6e6e"/>
        <SolidColorBrush x:Key="KinectPurpleBrush" Color="#ff52318f"/>
        <SolidColorBrush x:Key="KinectBlueBrush" Color="#ff00BCF2"/>
        <Style TargetType="{x:Type Image}">
            <Setter Property="SnapsToDevicePixels" Value="True"/>
        </Style>
        <Style TargetType="{x:Type CheckBox}" x:Key="SquareCheckBox" >
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="{x:Type CheckBox}">
                        <Grid>
                            <StackPanel Orientation="Horizontal"
Background="Transparent">
                                <Grid x:Name="SquareCheckBoxChecked">
                                    <Image x:Name="CheckedNormal"
Source="Images\CheckedNormal.png" Stretch="None" HorizontalAlignment="Center"/>
                                    <Image x:Name="CheckedHover"
Source="Images\CheckedHover.png" Stretch="None" HorizontalAlignment="Center"
Visibility="Collapsed"/>
                                </Grid>
                                <Grid x:Name="SquareCheckBoxUnchecked"
Visibility="Collapsed">
                                    <Image x:Name="UncheckedNormal"
Source="Images\UncheckedNormal.png" Stretch="None" HorizontalAlignment="Center"/>
                                    <Image x:Name="UncheckedHover"
Source="Images\UncheckedHover.png" Stretch="None" HorizontalAlignment="Center"
Visibility="Collapsed"/>
                                </Grid>
                                <TextBlock x:Name="SquareCheckBoxText"
Text="{TemplateBinding Content}" TextAlignment="Left"  VerticalAlignment="Center"
Foreground="{StaticResource KinectPurpleBrush}" FontSize="15"  Margin="9,0,0,0"/>
                            </StackPanel>
                        </Grid>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsChecked" Value="false">
                                <Setter Property="Visibility" Value="Collapsed"
TargetName="SquareCheckBoxChecked"/>
                                <Setter Property="Visibility" Value="Visible"
TargetName="SquareCheckBoxUnchecked"/>
                            </Trigger>
                            <Trigger Property="IsMouseOver" Value="true">
                                <Setter Property="Visibility" Value="Collapsed"
TargetName="CheckedNormal"/>
                                <Setter Property="Visibility" Value="Collapsed"
TargetName="UncheckedNormal"/>
                                <Setter Property="Visibility" Value="Visible"
TargetName="CheckedHover"/>
                                <Setter Property="Visibility" Value="Visible"
TargetName="UncheckedHover"/>
```

```
                                <Setter Property="Foreground" Value="{StaticResource
KinectBlueBrush}" TargetName="SquareCheckBoxText"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
    </Window.Resources>

    <Grid Name="layoutGrid" Margin="10 0 10 0">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <DockPanel Grid.Row="0" Margin="0 0 0 20">
            <Image DockPanel.Dock="Left" Source="Images\Logo.png" Stretch="Fill"
Height="41" Width="102" Margin="0 10 0 5"/>
            <Image DockPanel.Dock="Left" Source="Images\LogoUdg.png" Stretch="Fill"
Height="39" Width="96" Margin="0 10 0 5"/>
            <TextBlock DockPanel.Dock="Right" Margin="0 0 -1 0"
VerticalAlignment="Bottom" Foreground="{StaticResource MediumGreyBrush}"
FontFamily="Segoe UI" FontSize="18">Correcció postural esportiva utilitzant un
sensor Kinect</TextBlock>
            <Image Grid.Column="1" Source="Images\Status.png" Stretch="None"
HorizontalAlignment="Center" Margin="0 0 0 5"/>
        </DockPanel>
        <Viewbox Grid.Row="1" Stretch="Uniform"
HorizontalAlignment="Center"></Viewbox>
        <CheckBox Grid.Row="2" Style="{StaticResource SquareCheckBox}"
Content="Correcció postural activada" Height="Auto" HorizontalAlignment="Left"
VerticalAlignment="Center" Margin="0 10 10 10" Name="trackingOn" IsChecked="True" />
        <StatusBar Grid.Row="3" HorizontalAlignment="Stretch" Name="statusBar"
VerticalAlignment="Bottom" Background="White" Foreground="{StaticResource
MediumGreyBrush}">
            <StatusBarItem Padding="0 0 0 10">
                <TextBlock Name="statusBarText" Margin="-1 0 0 0">Activa per
corregir la activitat seleccionada!</TextBlock>
            </StatusBarItem>
        </StatusBar>

        <Image Name="Image" Width="640" Height="480" Margin="130,28,344,39"
Grid.Row="1" ImageFailed="Image_ImageFailed" HorizontalAlignment="Left" />

        <Slider Maximum="2" TickPlacement="BottomRight" TickFrequency="1"
IsSnapToTickEnabled="True" Name="slValue" Margin="839,89,89,0" Grid.Row="1"
Height="27" VerticalAlignment="Top" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="812,57,0,0" Name="textBlock1" Text="Selecció d'esport"
VerticalAlignment="Top" Width="222" FontSize="15" />

        <Slider Minimum="10" Maximum="30" TickPlacement="BottomRight"
TickFrequency="1" IsSnapToTickEnabled="True" Name="margeError"
Margin="812,468,72,55" Grid.Row="1" Value="15" />
```

```
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="812,435,0,0" Name="textBlock3" Text="Angles d'error permesos:"
VerticalAlignment="Top" Width="222" FontSize="15" />
        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="812,491,0,0" Name="EscalaMargeError" Text="10              15
20              25              30" VerticalAlignment="Top" Width="230"
FontSize="11" />


        <Image Grid.Row="1" Height="58" HorizontalAlignment="Left"
Margin="821,115,0,0" Source="Images\Running.png" Name="imageR" Stretch="Fill"
VerticalAlignment="Top" Width="56" />
        <Image Grid.Row="1" Height="58" HorizontalAlignment="Left"
Margin="905,115,0,0" Source="Images\Cuerda.png" Name="imageC" Stretch="Fill"
VerticalAlignment="Top" Width="56" />
        <Image Grid.Row="1" Height="58" HorizontalAlignment="Left"
Margin="995,115,0,0" Source="Images\Boxing.png" Name="imageB" Stretch="Fill"
VerticalAlignment="Top" Width="56" />


        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="812,214,0,0" Name="textBlock2" Text="Seguiment d'angles de les
articulacions" VerticalAlignment="Top" Width="222" FontSize="13" FontStyle="Italic"
/>

        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="855,253,0,0" Name="Coll" Text="Coll" VerticalAlignment="Top" Width="30"
FontSize="11" />
        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="855,280,0,0" Name="ED" Text="Espatlla dreta" VerticalAlignment="Top"
Width="73" FontSize="11" />
        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="1002,280,0,0" Name="EE" Text="Espatlla esquerre" VerticalAlignment="Top"
Width="86" FontSize="11" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="855,309,0,0" Name="CD" Text="Colze dret" VerticalAlignment="Top" Width="76"
FontSize="11" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="1002,309,0,0" Name="CE" Text="Colze esquerre" VerticalAlignment="Top"
Width="76" FontSize="11" />
        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="855,333,0,0" Name="MD" Text="Canell dret" VerticalAlignment="Top" Width="73"
FontSize="11" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="1002,334,0,0" Name="ME" Text="Canell esquerre" VerticalAlignment="Top"
Width="81" FontSize="11" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="855,361,0,0" Name="PD" Text="Cama dreta" VerticalAlignment="Top" Width="88"
FontSize="11" />
        <TextBlock Grid.Row="1" Height="19" HorizontalAlignment="Left"
Margin="1002,361,0,0" Name="PE" Text="Cama esquerra" VerticalAlignment="Top"
Width="93" FontSize="11" />
        <TextBlock Grid.Row="1" Height="18" HorizontalAlignment="Left"
Margin="855,393,0,0" Name="GD" Text="Genoll dret" VerticalAlignment="Top" Width="97"
FontSize="11" />
        <TextBlock Grid.Row="1" Height="15" HorizontalAlignment="Left"
Margin="1002,393,0,0" Name="GE" Text="Genoll esquerre" VerticalAlignment="Top"
Width="100" FontSize="11" />
```

```
        <TextBlock Grid.Row="1" Height="15" HorizontalAlignment="Left"
Margin="1002,252,0,0" Name="Pelvis" Text="Pelvis" VerticalAlignment="Top"
Width="100" FontSize="11" />

        <TextBox Name="MostrarAngleColzeE"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,305,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleColzeD"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,305,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleEspatllaD"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,277,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAnglePelvis"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,249,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleGenollD"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,389,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleCamaE"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,361,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleCanellE"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,333,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleCamaD"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,361,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleCanellD"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,333,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleColl"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="812,249,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleEspatllaE"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,277,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <TextBox Name="MostrarAngleGenollE"  Grid.Row="1" Height="22"
HorizontalAlignment="Left" Margin="962,389,0,0" VerticalAlignment="Top" Width="37"
IsReadOnly="True" />
        <CheckBox Grid.Row="1" Content="Usuari esquerrà" Height="Auto"
HorizontalAlignment="Left" VerticalAlignment="Top" Margin="812,192,0,342"
Name="usuariEsquerra" IsChecked="False" />




    </Grid>
</Window>
```

17 Manual d'usuari

17.1 Instal·lació

- Col·locar el sensor a sobre d'una taula o superfície plana on no pugui caure durant el seu ús.
- Instal·lar Kinect for Windows SDK de la pàgina oficial de Microsoft.
- Connectar el sensor Kinect a través d'un port USB 2.0 a l'ordinador personal.
- Executar el arxiu binari CorreccióEsportiva.EXE.

17.2 Requisits mínims

- Dispositiu Microsoft Kinect for Windows.

Ordinador Personal:

- Processador de 2GHZ o més ràpid.
- Targeta gràfica amb 1GB de RAM dedicada i suport per DirectX 9.0c.
- Connector USB 2.0 disponible.

Sistema operatiu:

- Microsoft Windows 7 o superior.

17.3 Manual

L'aplicació mostrarà per pantalla l'esquema de l'esquelet de l'usuari sobre un fons negre sempre que es trobi al camp de visió d'aquest. Si no es mostra cap quadre, probablement és a causa del fet que el sensor no està connectat.

L'aplicació també mostrarà a la part dreta els valors dels angles que formen les articulacions rellevants. Si no es mostra cap valor, probablement és a causa del fet que el sensor no està connectat o que l'usuari no es troba a dins del camp de visió del sensor.

Si apareix a la part inferior esquerre "No Kinect Ready!", és a causa del fet que no s'ha trobat cap sensor Kinect connectat.

Controls:

- Correcció postural activada: *Si està seleccionat, l'aplicació començarà a corregir postures depenent de l'esport seleccionat.*
- Selecció d'esport: *Aquest selector permetrà seleccionar entre les 3 activitats esportives de les quals es volen obtenir correccions.*
- Usuari esquerrà: *Si està seleccionat, quan s'estigui practicant l'activitat de boxa, l'aplicació tindrà en compte que la cama avançada és la dreta.*

- Angles d'error permesos: *Aquest desplaçador permet augmentar els angles a dins dels quals l'aplicació considera un moviment vàlid. El seu valor per defecte és 15, però augmentar-lo, augmentarà la permissibilitat de l'aplicació.*