Detecció gest d'una mà pel control d'un videojoc

Marina Bermúdez Granados,
Adrian Vargas Orellana, Laia Rubio Castro

Abstract — Aquest projecte final de l'assignatura de Visió per Computador se centra en la classificació de diferents gestos fets per una mà per controlar el comandament d'un videojoc 2D. Per tant, l'objectiu principal és fer una classificació ràpida i acurada, comprenent l'estat de l'art per entendre les pràctiques usuals i, en conseqüència, les més recomanades. Molts dels projectes similars investigats primer fan un tractament del *frame* o una extracció de característiques amb una segmentació per després fer la classificació. Les implementacions del videojoc són més variades, tot i que la majoria es desenvolupen en *Python*. Com a proposta i desenvolupament final del projecte, el grup primer ha generat un dataset propi per poder aplicar diferents algorismes de segmentació. Més tard s'ha fet una multitud de models, proves i validacions per determinar el millor. Els millors resultats ocorren amb els *frames* originals i les segmentacions *Otsu*, amb models com una *Support Vector Machine* (*SVM*) i un *K-Nearest Neighbour* (*KNN*). També s'han construït models més complexes com una *CNN* la qual ha mostrat resultats molt satisfactoris, tot per després arribar a l'objectiu final ja plantejat: El comandament d'un videojoc basat en els gestos d'una mà.

Keywords — Machine Vision, Sobel, Canny, Otsu, Segmentation, Historian of Oriented Gradients, Image Classification, KNN, SVM, CNNs, Gesture-based Games.

1 Introducció

En aquest document està detallat el desenvolupament del projecte final de l'assignatura Visió per Computador. El tema escollit tracta sobre el control dels moviments a un videojoc 2D amb diferents gestos d'una mà captats per una càmera. Per tant, el projecte es pot dividir en dues parts: La captació i classificació dels gestos de la mà, per després associar cada classe a una acció a fer al videojoc, com caminar o saltar uns obstacles.

La idea inicial va sorgir de vídeos que el grup va trobar sobre altres persones intentant jugar a videojocs sense cap comandament, només amb una càmera i una mica de programació. A partir d'aquests projectes, es va decidir intentar controlar un videojoc amb imatges captades i després ens vam decantar per fer el control amb els gestos de les mans.

L'objectiu principal del projecte se centrarà a fer una classificació prou acurada i ràpida com a poder passar aquestes dades a un videojoc, similar a com es faria normalment amb un comandament. Per poder fer seguiment del desenvolupament del codi, la documentació i les imatges resultant, el grup ha creat un repositori a Github[0].

2 Estat de l'art

Dins d'aquest projecte hi ha diverses tècniques que caldrà posar en pràctica. Primer cal estudiar com fer la detecció de la mà per poder captar i classificar les imatges corresponents. En acabar, s'haurà de passar aquesta classificació al videojoc per fer l'acció corresponent.

2.1 Detecció i Classificació d'una mà

Les tasques de detecció i classificació d'una mà segueixen normalment les mateixes tècniques. Tot i això, cal diferenciar entre classificar un gest o classificar els moviments necessaris per fer un gest. En aquest projecte es captarà *frame* a *frame* les imatges de la mà per detectar de manera binària si s'està fent un gest o no. Quan es detecti un gest per un mínim determinat de *frames*, llavors contarà com que si s'està fent.

Llavors, per detectar el gest d'una mà estàtica hi ha diverses implementacions que es poden fer servir. En primer lloc, es capta el *frame* i es processa perquè sigui més fàcil de tractar. Les implementacions manuals investigades passen la imatge a escales de grisos, per aplicar un threshold i tractar d'obtenir el contorn de la mà. També es poden treure les característiques en compte dels contorns, depenent de com es vol fer la classificació amb la informació extreta. Les llibreries trobades que s'ocupen del procés enter aconsegueixen pintar les guies de les falanges amb els punts importants detectats, per donar el percentatge de classificació més alt segons com estan col·locats. Tot i que només es té previst fer servir una mà, les llibreries també són capaces de diferenciar entre les mans dretes i esquerres. Aquest procés s'executa mitjançant models d'aprenentatge computacional.

2.2 Tècniques de Segmentació

Per la detecció de les mans dins d'una imatge o *frame*, es poden utilitzar tècniques de segmentació per separar-les del fons, i després analitzar i detectar la forma de la mà per saber quina acció fer. Es poden usar tècniques de segmentació clàssica, però hi existeix una altra opció per poder centrar el projecte només a l'anàlisi de les mans.

L'opció més simple per la segmentació de les imatges és fer servir *SAM* o *Segment Anything Model* [13], un model fet per Meta que com diu el nom, segmenta tot. Amb aquesta eina es podria aconseguir que aquest model segmenti les mans directament, per després poder fer una xarxa neuronal d'anàlisis la qual pot ser entrenada amb les nostres imatges generades i classificades. Tot i això, és un algorisme amb un cost molt elevat computacionalment, s'haurien de fer servir les tècniques clàssiques de segmentació.

2.3 Control del videojoc

Els vídeos mencionats anteriorment que han inspirat aquest projecte ja serveixen com a exemples sobre com controlar un

videojoc sense comandament. Tot i això, els primers feien servir detecció de postures o el moviment d'un cap a l'esquerra i la dreta. En fer una mica més de recerca, s'han trobat un terme per descriure els videojocs on es fan servir les mans pel seu control: *Gesture-based Games*. La majoria dels projectes trobats s'han desenvolupat en *Python*, tot i que el llenguatge de programació depèn en el videojoc.

3 Proposta

La nostra idea és utilitzar una base de dades creada per nosaltres, la qual constarà d'imatges de les nostres mans en diferents posicions així com, amb diferents fons. Això ens permetrà entrenar millor al nostre detector, ja que haurà de tenir en compte més posicions i possibles objectes o colors diferents de fons. Amb aquestes dades crearem un entorn de train i un de test per a detectar i captar les imatges per posteriorment fer la seva classificació.

A partir del dataset i per millorar la classificació, es provaran diversos mètodes de segmentació, extracció i descripció de característiques. La mateixa filosofia se seguirà amb els models de classificació, fer servir diferents opcions per poder escollir la millor. Pel que fa al llenguatge de programació, s'ha decidit escriure tot el codi amb *Python* perquè té moltes llibreries per totes les tasques necessàries per al desenvolupament del projecte. A més a més, el grup té més experiència en aquests entorns.

4 Experiments, resultats i anàlisis

El desenvolupament es divideix en diverses etapes, des de la creació del conjunt de dades fins a la classificació dels gestos i el comandament del videojoc. En cada apartat s'explicarà el codi desenvolupat i els resultats.

4.1 Creació dataset

La manera més efectiva de generar moltes imatges és gravant vídeos per poder extreure els seus *frames*. Per tant, cada membre del grup va fer vídeos curts fent els diferents gestos amb qualitats bones i dolentes, amb fons senzills i complicats, en diferents posicions i a diferents distàncies. Després només va fer falta llegir els *frames* i organitzar-los en diferents carpetes per cada gest. Per poder fer una possible adaptació a diferents resolucions, també s'ha forçat la conversió de totes les imatges a una mateixa resolució fixada. Es poden veure exemples dels resultats obtinguts a la Figura 1.




Fig. 1. Exemples dels diferents frames obtinguts.

A partir dels 50 vídeos gravats s'han pogut generar en total 2120 imatges pel primer signe, 2080 pel segon, 2232 amb el tercer, 1964 pel quart i finalment, 2142 pel cinquè. A l'apartat de classificació no es podrà utilitzar tot el dataset per problemes amb la memòria necessària per guardar totes les imatges a la vegada, normalment es faran les proves amb un quart o amb la meitat del dataset.

4.2 Segmentació

Per a la segmentació s'han utilitzat diversos mètodes que després es podran comparar a la seva posterior classificació. Dintre dels possibles algorismes de segmentació, s'han triat els que s'han considerat més adients per al projecte: *Sobel*, *Canny*, *Otsu* i segmentació per colors.

Sobel treballa sobre l'eix de les x i l'eix de les y per separat per posteriorment calcular la magnitud i direcció del gradient, completant així la segmentació. També s'ha aplicat una erosió per millorar els resultats en finalitzar l'algorisme. S'ha obtingut una bona segmentació en general, menys per les imatges amb fons més difícils on no ha sigut capaç d'obtenir un bon resultat.

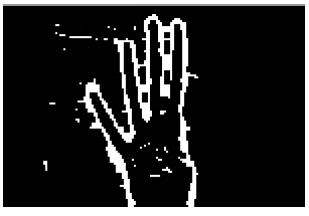

Fig. 2. Exemple segmentació amb Sobel

L'algorisme de *Canny* té com a objectiu detectar cantonades, les quals utilitza per identificar els diferents objectes de la imatge i així aconseguir la segmentació de les mans. Per a fons fàcils assoleix una bona segmentació, mentre que per a fons més complexos no és capaç d'aïllar les cantonades de la mà de les vores de la textura del fons.

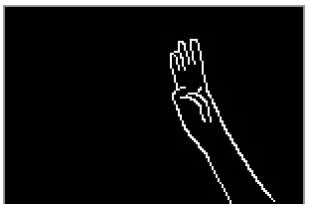

Fig. 3. Exemple segmentació amb Canny

L'algorisme d'*Otsu* segmenta els punts del *frame* basant-se en un paràmetre únic per a cada imatge. A causa d'aquesta metodologia, s'obtenen els millors resultats tot i que no aconsegueix la millor segmentació per a certes imatges. Un canvi notable respecte als altres mètodes és que aquest és capaç de realitzar una bona segmentació per a fons difícils. Per acabar de millorar els resultats, s'ha aplicat una erosió final.

Fig. 4. Exemple segmentació amb Otsu

Per a realitzar la segmentació per colors, primer s'ha passat la imatge a HSV i s'ha aplicat a la imatge original una màscara que delimita un rang per a cada color amb l'objectiu de segmentar el color de pell de la resta de colors. En comparació amb la resta d'algorismes, aquesta té la pitjor segmentació perquè el rang de color de pell varia per a cada imatge. A més a més, molts colors de fons són molt semblants al color de pell.

Fig. 5. Exemple segmentació per colors

4.3 Classificació

Amb les imatges segmentades, només queda fer la classificació i contrastar els resultats. S'han escollit 6 models diferents de probar: Un *K-Nearest Neighbours* (*KNN*) amb 5 veïns per començar, una regressió logística, un *Gaussian Naive Bayes*, un *Decision Tree*, un *Random Forest* i una *Support Vector Machine* lineal (*SVM*). Cada model s'ha provat amb totes les segmentacions, executant diferents versions per l'entrenament i test amb les imatges senceres aplanades o amb el descriptor de característiques *Histogram of Oriented Gradients* (*HOG*).

De totes les opcions, les millors classificacions han sorgit amb les imatges originals i les segmentades amb *Otsu*. El millor resultat a ambdues versions en les mètriques de precisió, *recall* i *f1-Score* és del model *SVM* lineal amb un 97% en tot utilitzant la imatge sencera aplanada. El segon model amb les millors mètriques és *KNN* amb un 93% en tot, fent servir la imatge sencera aplanada i els descriptors de *HOG*. La qualitat de les classificacions es pot veure a les matrius de confusió a les Figures 6 i 7.

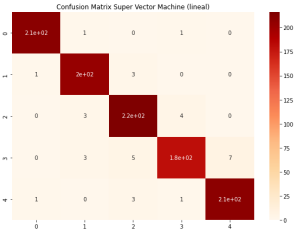

Fig. 6. Matriu Confusió SVM

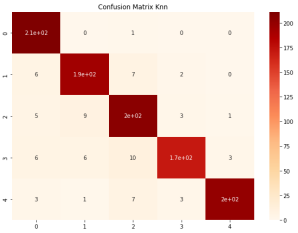

Fig. 7. Matriu Confusió KNN

Les segmentacions *Sobel* i *Canny* arriben a resultats més propers al 70% a les millors classificacions, mentre que la segmentació per colors només arriba al 38% a les mètriques *recall* i *f1-Score*. Pel que fa als temps d'execució, la *SVM* és el model que triga més i el *Naive Bayes* és el que triga menys, amb una de les pitjors classificacions a totes les segmentacions.

Tot i que el temps de predicció és molt important en el context d'aquest projecte, és una mesura que depèn en gran part en el *hardware* disponible per la seva execució. Per tant, la prioritat està en assegurar i validar els resultats de *KNN* i *SVM*. Abans de començar amb les validacions, cal trobar els millors hiperparàmetres per *KNN*, concretament la millor quantitat de veïns. Fins ara s'han fet les proves amb una k=5 i una divisió d'entrenament i test del 80% i 20%. Mantenint aquesta mateixa divisió, s'han mesurat les mètriques resultants en canviar el valor k. Les millors mètriques són k=2 si es fa servir la imatge sencera i k=3 si es fa servir els descriptors *HOG*, com es pot comprovar a les gràfiques de les Figures 8 i 9.

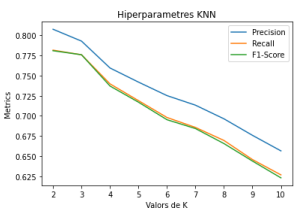

Fig. 8. Gràfica mètriques KNN

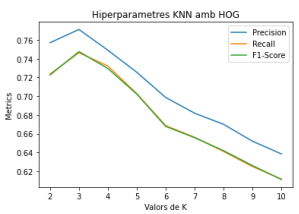

Fig. 9. Gràfica mètriques KNN amb descriptors HOG

La validació dels models s'ha basat en el mètode *K-Fold* amb 10 *splits*, canviant les divisions d'entrenament per veure el seu rendiment. Primer s'han fet les comprovacions amb *KNN* k=2 fent servir la imatge sencera aplanada i els descriptors *HOG*. La seva millor execució té una mitjana de mètriques propera al 90% utilitzant la imatge sencera aplanada i la meitat del dataset per l'entrenament, com es veu a la Figura 9. Els descriptors de *HOG* tenen resultats lleugerament pitjors a la

Figura 10. Les comprovacions amb *SVM* segueixen el mateix patró, arribant al 92% amb la imatge sencera a la Figura 9. Tot i això, els descriptors de *HOG* en aquest model funcionen millor amb un 60% del dataset dedicat a l'entrenament com es pot veure a la Figura 10.

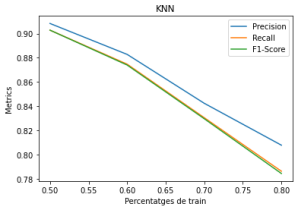

Fig. 9. K-Fold KNN

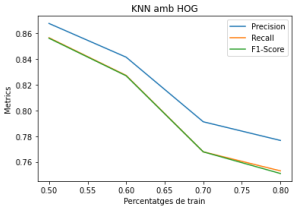

Fig. 10. K-Fold KNN amb descriptors HOG

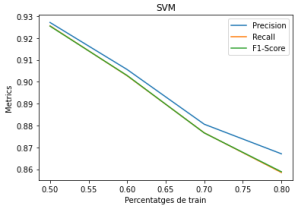

Fig. 11. K-Fold SVM

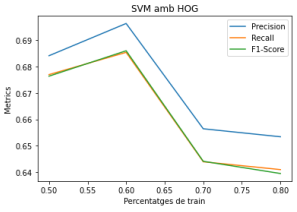

Fig. 12. K-Fold SVM amb descriptors HOG

4.4 Xarxes neuronals

Les Xarxes Neuronals són una eina molt útil per la classificació, tot i que tenen desavantatges com que són uns models de caixa negra. Tot i això, s'ha decidit provar aquest tipus de model, ja que normalment tenen molt bons resultats de temps de predicció.

Primer s'han fet varies Xarxes Neuronals Simples entrenades amb 50 èpoques amb diferents tipus de funcions de pèrdua o *Loss Function*, totes amb les mateixes característiques: Una capa oculta de tipus Densa de 150 neurones amb activació *Relu*, una d'entrada de tipus Flatten amb entrades de mida (100,150,3) i una sortida amb una capa Densa amb 5 sortides possibles, una per cada signe. P. Les *Loss Functions* que s'han provat són: *Mean Squared Error*, *Mean Absolute Percentage Error*, *Categorical CrossEntropy* i *Mean Absolute Error*.

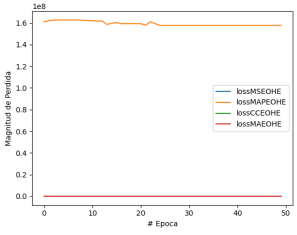

Fig. 13. NN *Loss* amb imatges originals

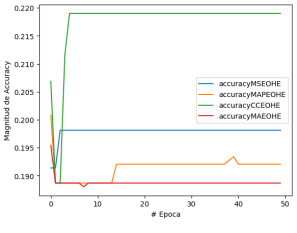

Fig. 14. NN accuracy amb imatges originals

Amb aquestes condicions i utilitzant les imatges originals, s'han aconseguit els resultats de les Figures 13 i 14. D'aquestes gràfiques podem treure la conclusió que la millor *Loss Function* pel nostre model és la *Categorical Crossentropy*. Tenint en compte això, s'ha entrenat un model de *CNN* (*Convolutional Neural Network* o Xarxa Neuronal Convolucional) amb les següents característiques: 2 Capes de Convolució amb el mateix input que abans de 32 i 64 neurones, 3 Capes ocultes de tipus Densa de 150 neurones amb funció d'activació *Relu*, 1 capa Densa de sortida amb 5 neurones, i 100 èpoques d'entrenament. Amb aquesta configuració s'han assolit resultats molt bons, amb precisions de fins al 90% amb diferents imatges d'entrada, com a la Figura 15.

Fig. 15. Prediccions CNN amb les imatges originals

Les classificacions de la *CNN* amb totes les combinacions anteriors han arribat a igualar i fins i tot millorar els resultats de la *SVM* i *KNN* a les segmentacions *Sobel* i *Canny*. El seu millor resultat també ocorre a la segmentació *Otsu* amb un 93% i una matriu de confusió mostrada a la Figura 16. La validació de la *CNN* s'ha fet igual que els models a la visió clàssica del problema, amb un *K-Fold* de 10 *splits* i diferents divisions d'entrenament i test. A comparació dels models anteriors, la *CNN* sempre millora per percentatges cada vegada més grans d'entrenament. El seu màxim amb els percentatges provats és de 90% amb una divisió d'entrenament i test del 80% i 20% com es pot veure a la Figura 17.

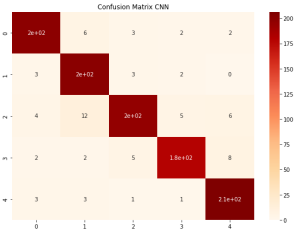

Fig. 16. Matriu Confusió CNN

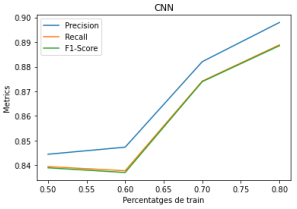

Fig. 17. K-Fold CNN

4.5 Videojoc

Un cop ja s'han provat tots els models, segmentacions i opcions possibles, es pot començar amb la implementació del joc tenint ja una idea de les millors combinacions. Com desenvolupar un videojoc és una tasca molt complicada, el

videojoc per aquesta part s'ha basat en un dels projectes trobats durant la recerca de referents. L'original es pot trobar a la bibliografia referenciat[2]. Amb aquesta base, s'ha adaptat el codi al funcionament dels models estudiats en comptes de llibreries externes, modificant les parts corresponents per tal que el jugador pugui saltar amb els signes parells indicats per càmera. En provar el videojoc amb la *CNN* final entrenada amb 100 èpoques i les imatges originals, el joc va bastant fluid i funciona molt bé, tot i que no és perfecte. Encara es podrien provar d'aplicar les segmentacions i models ja implementats. Així i tot, ja té un funcionament prou correcte com es pot comprovar a la Figura 18..

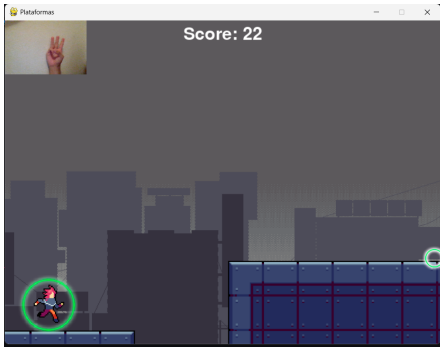

Fig. 18. Demostració del funcionament del videojoc

5 Conclusions

Durant l'execució d'aquest projecte s'ha consolidat com fer servir les diferents tècniques de visió per computador per resoldre un problema real de classificació. S'han implementat des de segmentacions per l'extracció de característiques fins a descriptors, classificacions i models més complexes com una *CNN*. Tot i això, la part més complicada va ser el plantejament inicial, decidir exactament que fer i els objectius. A l'estat de l'art, el grup es va informar de moltes maneres diferents d'aconseguir completar la idea inicial: Controlar el comandament d'un videojoc amb moviment per càmera. A més a més, cada exemple varia segons el tipus de videojoc i com es vol detectar el moviment. Al final, es va decidir fer servir la mà. A partir d'aquest moment, només va ser fer divisió de tasques per la segmentació, classificació, la *CNN* i el videojoc.

A partir de la solució final desenvolupada es podrien afegir moltes millores, sobretot al procés de classificació. A l'apartat de l'estat de l'art es van trobar llibreries com a *MediaPipe* [10] la qual està especialitzada específicament per detectar gestos de les mans fent servir punts de control amb les falanges. En ser una llibreria serà més ràpida i acurada que la implementació manual. Tot i això, el grup ha volgut intentar desenvolupar un codi funcional completament propi. La mateixa filosofia s'ha aplicat amb el dataset, tot i que s'ha trobat el *HAnd Gesture Recognition Dataset* (*HAGRID*) [11], s'ha tractat de generar un dataset personal. D'aquesta manera s'ha après com fer totes les etapes necessàries. Altres millores més relacionades al videojoc podrien ser afegir més signes o mecàniques.

Bibliografia

[0] "Hand-Clasification-VC-Project: Classification of Hand Gestures to Control a 2D Videogame." *GitHub*, https://github.com/adriend1102/Hand-Clasification-VC-Project.

[1] Tech, R. (2023). Uso Python e IA para jugar sin control y se rompe la silla [Video]. In *YouTube*. https://www.youtube.com/watch?v=vroIHqqz3v0

[2] Tech, R. (2023). Intento hacer juegos con Python e IA y me explota la cabeza [Video]. In *YouTube*. https://youtu.be/_BjL6W71mWY

[3] Aslananuragi. (2021, June 16). *Gesture controlled video game*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/gesture-controlled-video-game/

[4] atul, kang &. (2019, March 4). *Snake Game using Tensorflow Object Detection API*. TheAILearner. https://theailearner.com/2019/03/04/snake-game-using-tensorflow-object-detection-api/

[5] Wang, E., & Arif, M. (n.d.). *Gesture controlled gaming*. Devpost. Retrieved April 28, 2023, from https://devpost.com/software/gesture-controlled-gaming

[6] Haria, A., Subramanian, A., Asokkumar, N., Poddar, S., & Nayak, J. S. (2017). Hand gesture recognition for human computer interaction. *Procedia Computer Science*, *115*(115), 367–374. https://doi.org/10.1016/j.procs.2017.09.092

[7] Haroon, M., Altaf, S., Ahmad, S., Zaindin, M., Huda, S., & Iqbal, S. (2022). Hand gesture recognition with symmetric pattern under diverse illuminated conditions using artificial neural network. *Symmetry*, *14*(10). https://doi.org/10.3390/sym14102045

[8] Pol, S., Pagade, P., & Pati, D. (2017). Gesture Recognition Based Video Game Controller. *International Research Journal of Engineering and Technology (IRJET)*, *04*(11). https://www.irjet.net/archives/V4/i11/IRJET-V4I11182.pdf

[9] MediaPipe. (2023, January 19). *Gesture recognition task guide*. Google Developers. https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer

[10] hukenovs. (n.d.). *GitHub - Hukenovs/hagrid: HAnd Gesture Recognition Image Dataset*. GitHub. Retrieved April 28, 2023, from https://github.com/hukenovs/hagrid

[11] OpenMMLab. (n.d.). OpenMMLab. Retrieved April 28, 2023, from https://openmmlab.com/

[12] Meta AI. (n.d.). Segment Anything. Retrieved April 28, 2023, from https://segment-anything.com/