Universität für Musik und darstellende Kunst Graz

IEM - Institut für Elektronische Musik und Akustik

Projekt – SS2008

Ganzkörper Gestenerkennung

Bericht über die Implementierung des Algorithmus

von
Miha Ciglar
miha@mur.at

Einleitung

Der Algorithmus wurde in PD (Pure Data) implementiert, wobei auf Windows und mit dem PD extended [1] Installationspaket gearbeitet wurde.

Es wurde ein spezielles Konzept der Ganzkörper-Gestenerkennung verfolgt, mit dem Ziel, eine Unempfindlichkeit des Erkennungsalgorithmus auf die Ausdehnung der Geste im Raum, sowie auf die zeitliche Interpretation der Geste zu erreichen. Details zum Konzept und zur generellen Herangehensweise sind in [2] und [3] ausführlich beschrieben.

Für die Datengewinnung wurde das *Vicon* Motion tracking System eingesetzt, mit dem, die Koordinaten bzw. die Trajektorien von vier, an den Extremitäten-enden angebrachten Markern, berechnet werden konnten.

Datenverarbeitung - (Implementation in PD)

main.pd

[pd osc_in]:

Die Rohdaten des Vicon Systems kommen über das OSC Protokoll in Echtzeit in den PD Patch, und werden im Mudul **[pd osc_in]** empfangen. Bei den Input-Daten handelt es sich konkret um einen 12-dimensionalen Vektor (3 räumliche Koordinaten von 4 Punkten).

[pd euklidian distance]:

- **[pd dist a - b]**
- **[X Y Z]**

Im folgendem Modul **[pd euklidian distance]** werden die 6 Distanzen zwischen diesen 4 Punkten kontinuierlich berechnet. Dies geschieht in den Sub-Modulen: **[pd dist a - b]**. Zuvor jedoch wird in den Sub-modulen **[X Y Z]**, die (0,0,0) Koordinate gefiltert, die immer auftritt wenn die Marker vom Tänzer für einen Augenblick verdeckt werden und daher für kurze Zeit verloren gehen.

[pd pre-proces_-_extract_direction]:

Die 6 Werte, die die veränderlichen Distanzen zwischen den 4 Punkten beschreiben, werden zum nächsten Modul **[pd pre-proces_-_extract_direction]** weitergeleitet. Dort folgt eine Tiefpassfilterung und die Bestimmung der tendenziellen Entwicklung der einzelnen Distanz (*zunehmend, abnehmend* oder *unverändert*, i.e.: 1,-1,0).

- **[lpf]** - Tiefpassfilter – Berechnung des Durchschnittswertes mehrerer Samples (50% Überlappung)
- **[d]** - erste Ableitung
    - **[Z^1]** – Delay - (1 Sample)
- **[det_dir]** - Vorzeichenbestimmung / Richtungsbestimmung - (1,-1,0)

[pd gesture_follower_____]:

Die einzelnen Dimensionen des 6-dimensionalen Distanzvektors, können ab hier nur mehr 3 verschiedene Werte annehmen, nähmlich: (1,-1 ind 0). Das Modul: **[pd gesture_follower_____]** beherbergt nur sub-Module der Art: **[train_compare_gesture_model],** die jeweils eine Geste repräsentieren, die es zu erkennen gilt.

[train_compare_gesture_model]:

In diesem sub-Modul werden also die Daten einer einzelnen aufgenommenen Geste gespeichert und später auch mit dem live-Input verglichen. Am Output erscheint, der Prozentsatz der erkannten Geste.

- **[pd rec_ctl]** – Verarbeitung der Controlltrigger (Aufnahme - start/stop/clear/enable)
- **[pd record_gesture_+_determine_clusters_____]**

In diesem sub-Modul werden die gesten-Daten aufgenommen und in zeitlichen Clustern zusammengefasst.

- **[6D-to-1D-Vector] –** umwandeln des 6- zum 1-dimensionalen Vektor
- **[cluster_buff]** **–** Clustering

[cluster_buff]:

- **[pd watchdog]:**

Überwachung des Zeitintervalls zwischen den, am Input erscheinenden (diskretisierten) Daten, bzw. Distanzvektoren, die keine konstante Samplingrate mehr aufweisen, da ein nuer Zustandswektor nun immer nur dann generiert wird, wenn eine Richtungsänderung in mindestens einer Dimension vorkommt. Ab einem bestimmten Zeitwert ($2) wird ein neuer Cluster gebildet, bzw. eine Clustergrenze markiert. Gleichzeitig gibt es hier auch eine Bedingung für die minimale Clustergröße (minimale Anzahl der Vektoren im Cluster (z.B: 3))

- **[pd init]** **-** Initialisierung der temp-Buffers
- **[pd write_to_temp_buffers]:**

Die Daten werden zuerst in 2 Temporäre Buffer geschrieben: *cluster_buffer* (enthält die – nun eindimensinalen – Zustandsvektoren) und *priority_buffer* (enthält die Information über die Anzahl bzw. Häufigkeit des Vorkommens jedes Zustandsvektors im betrefenden zeit-Cluster), die nach dem Empfang des Watchdog Zeichens, für das setzen von Clustergrenzen entleert werden, bzw: deren Daten in die Main-Buffer geschrieben werden.

- **[pd set_priority]:**

Überwacht die Häufigkeit des Auftretens bestimmter Zustände im zeit-Cluster und schreibt den Wert (die Anzahl der gleichen Zustände) in den priority-Buffer. Der cluster-Buffer verzeichnet nur das erste Auftreten eines bestimmten Zustands. Die Adresse des häufigkeits-

Wertes im priority-Buffer stimmt mit der Adresse des betreffenden Zustandes im cluster-Buffer überein.

- **[pd copy_to_main_buffers]**

Kopiert den Inhalt der temporären cluster- und priority-Buffer in die main-Buffer, nach jeder Meldung des Watchdog. Die Enden, bzw. die Grenzen zwischen den Clustern werden mit dem Wert „-50“ im *main_buffer_c* und mit „0“ in dem *main_buffer_p* markiert.

- **[clear_buff]** – Initialisiert beide main-Buffer vor jeder Aufnahme
- **[pd r/w_switch_____]:**

Schaltet um zwischen *record* und *compare* Modus. Im ersten werden die Daten aus den temporären Clustern in die main-buffer geschrieben, im zweiten Modus (*compare*), werden die Daten an andere Module weitergeleitet.

- **[W-C]** – einfacher Schalter (Write / Compare)

[pd gesture_recognition_____]:

In diesem Modul wird der (geclusterte) live Input, mit den aufgenommenen Clustern in den main-Buffern verglichen. Verglichen wird zunächst der erste aufgenommene Datencluster, der den Anfang der aufgenommenen Geste darstellt. Sobald eine Übereinstimmung mit dem live-input-Cluster vorliegt, wird der Vergleich des nächsten live-input-Clusters mit dem zweiten aufgenommenem Cluster vorgenommen, usw. Sobald die beiden Cluster zu sehr voneinander abweichen, wird zunächst ein Vergleich mit der Umgebung des gerade verglichenen Clusters vorgenommen (z.B: + 3 und - 2 benachbarte Cluster) – was als Time-warping bezeichnet werden kann. Wenn immer noch keine Übereinstimmung gefunden wird, wird der Vergleich abgebrochen, und das System kehrt zurück in den ersten Zustand – (Überwachung und Vergleich des Ersten Clusters mit dem live Input).

Der priority-Wert, der sich üblicherweise von 1 bis 4 bewegt wird mit 1000 Multipliziert und zum Zustandswert der von (0 bis 3^6 bzw. 729) reicht dazugezählt, um später mit den [list append] und [sort] Objekten eine Sortierung nach Größe vorzunehmen. Zum Schluss wird eine Übereinstimmung in den am häufigsten vorkommenden Zuständen höher bewertet als eine Übereinstimmung in den restlichen Zuständen.

- **[pd compare_cluster_ states__]:**

In diesem sub-Modul werden die einzelnen Zustände im Cluster – alle gegen alle – verglichen, so dass, deren Reihenfolge im Cluster keine Rolle mehr spielt.

- **[unpack+filter_16_____]:**

Entpackt die sortierten Listen und filtert die mit „-50“ markierten Zustände, die die Cluster-Grenzen symbolisieren.

- **[pd count+compare]:**

Zählt die Anzahl der Zustände im live-Input Cluster und die im aufgenommenem Cluster, die gerade miteinanger verglichen werden. Es bestimmt weiters die kleinste gemeinsame Anzahl an Zuständen, um das übereinstimmungs-Resultat damit normieren zu können.

- **[pd compare_clusters+count_matches] -** Vergleich der einzelnen Zustände (alle gegen alle) (*aufgenommen* gegen *live*)
- **[cross_compare]:**

Die einzelnen Zustandswerte werden wieder rücktransformiert (in die ursprünglichen 6 Dimensionen) – **[1D-to6D-Vector]** und schließlich jede Dimension einzeln mittels Subtraktion auf eine Abweichung untersucht. Das Ideale Resultat dieses Moduls ist „0“. Eine einzelne Dimension kann nur die Werte (1,-1 oder 0) annehmen, also ist das schlechteste Resultat (alle Dimensionen weichen maximal voneinander ab) gleich „12“. Durch setzen einer Grenze ($1) kann man die maximal erlaubte Abweichung definieren. Falls diese Grenze nicht überschritten wird liefert das Modul am Ausgang einen „Bang“. Die Gewichtung der

oben erwähnten dominanten Zustände (mit höchster Vorkommenshäufigkeit) wird später, z.B. mittels Verdopplung des „Bangs“ erreicht. Alle Treffer werden zusammengezählt und bilden das Resultat des cluster-Vergleichs.

- **[pd check neighbors]:**

Wenn keine Übereinstimmung vorliegt, wird hier die Umgebung des aufgenommenen Cluster auf eine mögliche Übereinstimmung untersucht (dies kann beliebig variieren, z.B. bis + 3 und - 2 benachbarte Cluster). Bei der Evaluierung dieses Resultats wird im Modul: **[pd compare_clusters+count_matches]** zum Teil auch das Ergebnis des original-Vergleichs berücksichtigt. (siehe Kommentar zum Code: „sequential dependency“).

- **[time warp]:**

Sobald es durch einen nachbar-Vergleich zu einem positiven Ergebniss kommt (Übereinstimmung), muss der momentane (also Tatsächliche) Stand der Gestenerkenntnis aufgefrischt werden. Dies geschieht also im Modul: **[time warp].**

Schlusskomentar

Auf wichtige Details der Implementierung weisen Kommentare hin, die direkt beim program-Code dazugeschrieben sind.

Referenzen:

[1] - http://puredata.info/downloads

[2] - Ciglar, M. “3rd. Pole – a composition performed via gestural cues”, in Proc. of the International Conference on New Interfaces for Musical Expression 2008 (NIME 08), Genova, Italy

[3] - Ciglar, M. “A full-body gesture recognition system and its implementation in the composition 3r. Pole” in Proc. of the International Computer Music Conference 2008 (ICMC 08), Belfast, Ireland