Dokumentation VR-Visualization

Virtual Reality, Master Informatik

Matthias Haselmaier, Andreas M. Brunnet

Vorwort

Im Rahmen der Veranstaltung *Virtual Reality* wurde eine VR-Anwendung auf Basis der Unity 3D-Engine entwickelt, mit deren Hilfe eine Betrachung von Machine Learning Daten im Raum ermöglicht wird. Hierbei hat der Anwender die Möglichkeit innerhalb der als Scatterplots visualisierten Daten umher zu wandern. Darüber hinaus lässt sich ein für den Nutzer interessanter Datenpunkt mittels VR üblicher Selektiergestik auswählen. Die Attribute und Werte des selektierten Datenpunks werden relativ zu seiner Position dem Nutzer dargestellt. Über das Hervorheben des selektierten Punktes in sämtlichen Scatterplot-Matrizen hat der Anwender direkte Übersicht über die Verteilung des Punktes innerhalb des Datensets.

Allgemeine Informationen

Software und Bibliotheken

Verwendete Software und Bibliotheken

MiddleVR ([mvr]): Dient der Entkopplung von VR-Hardware und der Applikation. Als Middleware abstrahiert MiddleVR Eingaben durch Hardware wie der HTC Vive Pro und stellt diese auf Seite der Anwendung als verallgemeinerte Schnittstelle zur Verfügung.

Während der Navigation in UI Komponenten fiel auf, dass Tasteneingaben nicht *debounced* werden. So werden beispielsweise Checkboxen schwer steuerbar. Auch andere UI Komponenten waren unter diesen Bedingungen nicht zufiedenstellend nutzbar. Entsprechend wurde innerhalb der zuständigen MiddleVR Klasse *VRCanvas* die Abfrage von Tastenevents auf *Toogled* umgestellt, um so ein *pressed* / *released* Verhalten zu erhalten.

IATK ([iatk]): Das Toolkit stellt eine Reihe von Funktionalitäten für das Arbeiten mit Daten zur Verfügung. Unter anderem ist ein CSV-Parser enthalten, der im Gegensatz zu einem üblichen CSV-Parser Wertnormalisierungen unabhängig vom vorliegenden Datentyp vornimmt.

Es musste eine Anpassung am Parser vorgenommen werden. Dort wird in der Methode *GetOriginalValue* ein Float-Wert als Index für ein Array genommen. Hierbei wird die implizite Typkonvertierung und die daraus resultierende Abrundung zum Problem, da an diesem Punkt ein Runden zur nächsten Ganzzahl erfolgen sollte. Dieser Fehler wurde mittels *Mathf.Round* behoben.

Die Möglichkeiten des Frameworks zur Datenvisualisierung wurden nach ersten Tests mit dem Framework verworfen, da eine Eigenimplementierung der nötigen Visualisierungskomponenten mit den nötigen Interaktionsmöglichkeiten, wie der Datenpunktauswahl (siehe Kurzeinleitung, Abb. 1) somit umsetzbar wurde.

Verworfene Software und Bibliotheken

VRTK ([vrtk]): Wie auch MiddleVR dient VRTK der Entkopplung von Eingabegeräten und der

Applikation. Eine erste Integration in das Projekt verlief ohne größere Probleme, sodass das Testen der Anwendung mittels Maus und Tastatur (Modus: Simulation) möglich war. Probleme traten jedoch beim Versuch der Integration einer Vive-Pro Konfiguration auf, da VRTK seit einiger Zeit nicht mehr aktualisiert wird. Entsprechend kam es zu Komplikationen mit der aktuellen API der Vive-Pro (SteamVR, [svr]). Entsprechend wurde als alternative Lösung MiddleVR erfolgreich evaluiert und im Projekt statt VRTK integriert.

Benutzung

Die Anwendung benötigt zum Starten eine vorhandene MiddleVR Konfigurationsdatei. Die Voreinstellung des Pfades verweist auf die Datei am VR-Labor Rechner. Kann diese Datei nicht gefunden werden oder wird die Anwendung auf einem anderen System ausgeführt, muss diese mit folgendem zusätzlichen Argument gestartet werden:

Listing 1. MiddleVR Konfiguration laden

```
VR-Visualization.exe --config "path/to/my/config.vrx"
```

Hiermit wird der voreingestellte Pfad überschrieben und die angegebene Konfigurationsdatei stattdessen geladen.

Kurzeinleitung

Das Projekt kann über das GitHub-Repository bezogen werden. Folgendes beinhaltet das Repository:

- Dokumentation als .pdf, .html und dessen Sourcen
- Applikationssourcen
- Lauffähiger Build im entsprechenden Verzeichnis (Auf VR-Lab Rechner getestet)

Nach dem Starten findet sich der Nutzer im leeren 3D-Raum wieder.

Abbildung 1. HTC Vive Pro Controller

Für die UI Steuerung sind die *Menütaste* und der *Trigger* von Interesse (Bsp. Abb. 1). Nach Betätigen der Menütaste öffnet sich der Import Dialog (Abb. Abb. 2).

Abbildung 2. Dialog zum Datenimport

Der Dialog bewegt sich mit der Kamera des Anwenders. Somit lässt sich der Dialog an jeder Position in der Szene in eine angenehm zu bedienende Position bringen.

Dort kann die gewünschte CVS-Datei ausgewählt werden. Danach lässt sich über die Checkbox-Liste auswählen, welche Attributskombinationen in der Szene visualisiert werden sollen. Mit dem Import-Button schließt sich der Dialog und die Scatterplot-Matrix wird aufgebaut.

Nun kann der Anwender mit Hilfe des Steuerkreuzes oder Steuerpads (siehe Bsp. Abb. 1) zwischen den Scatterplots umher wandern (Abb. Abb. 3).

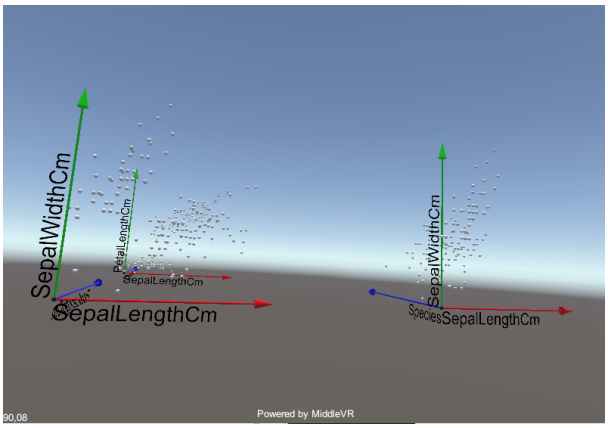


Abbildung 3. Scatterplot Matrix

Ist ein Datenpunkt von besonderem Interesse, kann dieser mit Hilfe des *Wands* und drücken des Triggers selektiert werden. Der Datenpunkt erscheint nun in roter Farbe und größerer Skalierung (Abb. Abb. 1).

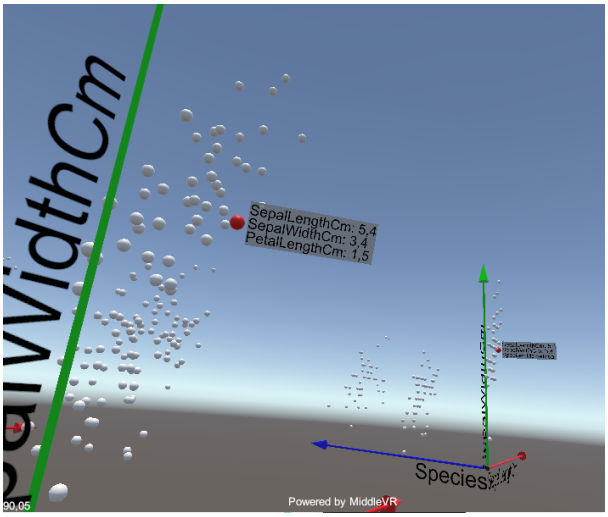
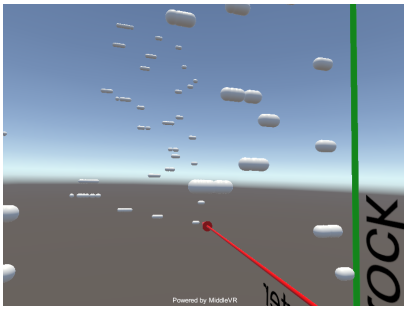


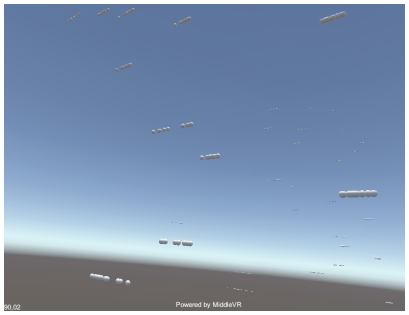
Abbildung 4. Selektion eines Datenpunktes

Das Hervorheben des Datenpunkts findet hierbei in jedem Scatterplot statt, der diesen Punkt beinhaltet. Zusätzlich zum visuellen Hervorheben des Punktes wird relativ zu dessen Position und stets zum Anwender gerichtet eine Liste der drei Attribute und Werte des Datenpunktes dargestellt.

Sollten dichtere Punktwolken vorliegen und die Auswahl eines Punkes deshalb zu schwierig sein, lässt sich über das Import Menü die größe der Datenpunkte (Abb. Abb. 2) in Echtzeit ändern. Die zuständigen Buttons finden sich unter dem Importieren-Button (Abb. Abb. 2). Umgekehrt lassen sich die Punkte auch vergrößern um beispielsweise bei wenigen Punkten eine bessere Visualisierung zu erzielen.

Skalierung von Datenpunkten





Aufgewendete Zeit

VIEL

Architektur der Anwendung

Zur Visualisierung der Daten stellt die VR-Anwendung vier Komponenten zur Verfügung, welche hierachisch aufgebaut sind. Die *Visualizer*-Komponente stellt die Schnittstelle zwischen der UI und der internen Datenstruktur dar. Sie ist dafür zuständig, die Daten aus einer CSV-Datei einzulesen, der UI die nötigen Informationen bereitzustellen und die vom Benutzer gewählte *ScatterplotMatrix* zu erstellen. Die *ScatterplotMatrix*-Komponente wird als Kind des *Visualizers* erstellt. Sie ist für das Erzeugen und Positionieren der einzelnen *Scatterplots* verantwortlich. Die *Scatterplot*-Komponente erstellt ein 3D-Diagramm, in dem die drei übergebenen Dimensionen für alle *DataPoints*

gegeneinander aufgetragen werden. Die *DataPoint*-Komponente stellt einen Datenpunkt als Kugel im *Scatterplot* dar. Wie die Szene nach dem Importieren einer CSV-Datei aussieht, ist auf Abb. 4 gezeigt. Im Folgenden werden die Komponenten genauer beschrieben.

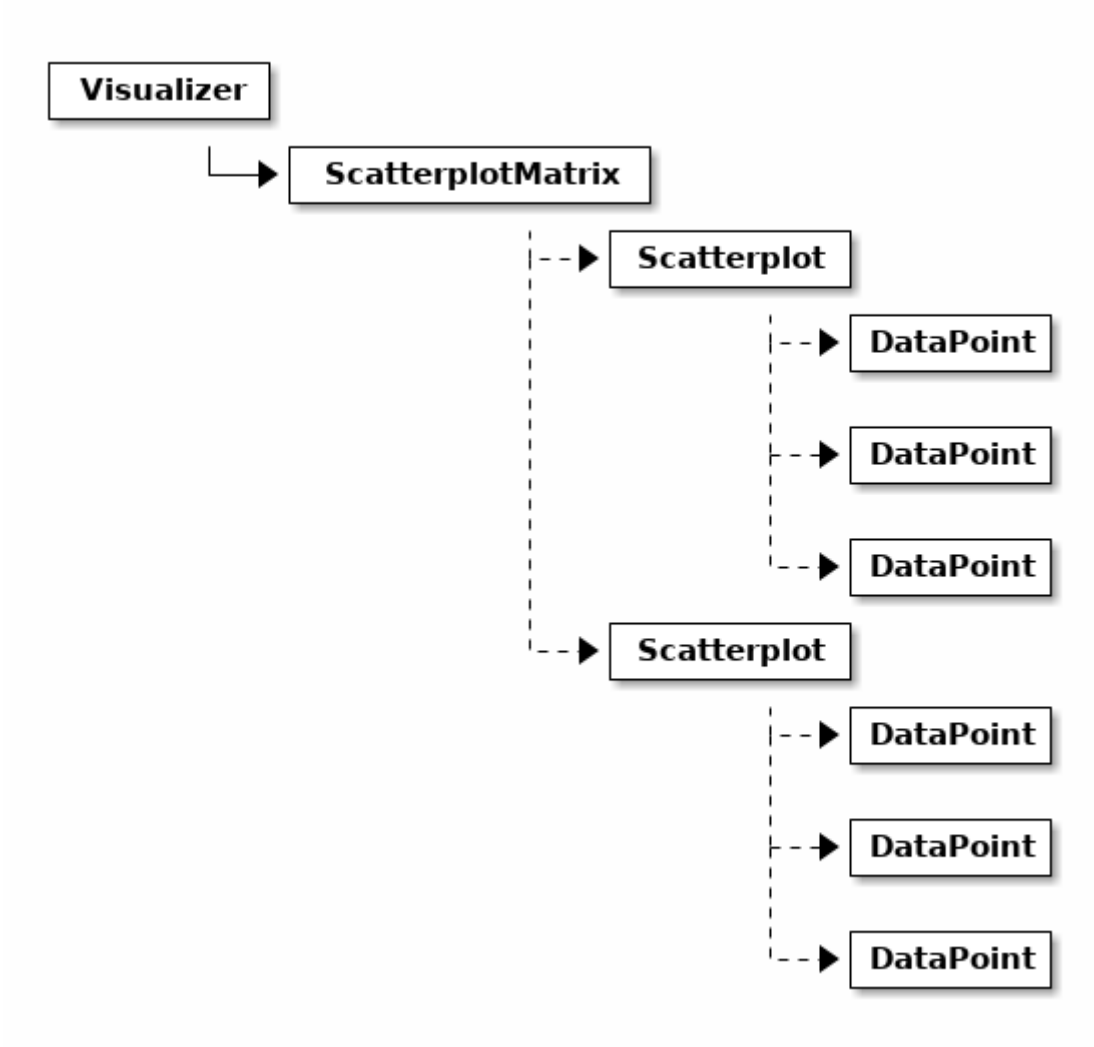


Abbildung 5. Szene

Visualizer

Die Schnittstelle des *Visualizer* besteht drei Methoden und einem Attribut. Die Methoden *LoadDataSource(TextAsset)* wird vom *ImportDialog* aufgerufen, wenn der Benutzer im DropDown-Menü eine neue CSV-Datei auswählt. Neben dem eigentlichen Laden der CSV-Datei werden zusätzlich alle möglichen 3er-Kombinationen der in der Datei enthaltenen Dimensionen gebildet. Hierzu wird *CalculatePossibleScatterplots()* aufgerufen, welche ein zweidimensionales *int*-Array zurück gibt. Die erste Dimension dieses Arrays stellt dabei die einzelnen Kombinationen der Dimensionen der CSV-Datei dar. Die zweite Dimension hat die Größe 3 und besteht aus den drei Indizes der jeweiligen Dimension der CSV-Datei. Der *ImportDialog* erhält die 3er-Kombinationen als *string* -Array über die Methode *GetPossibleScatterplots()*. Somit können dem Benutzer die möglichen *Scatterplot*-Varianten angezeigt werden. Wird der *Importieren*-Button angeklickt, ruft der *ImportDialog* die Methode *CreateScatterplotMatrix(int[])* auf. Über den Parameter werden die Indizes der vom Benutzer gewünschten *Scatterplots* übergeben. Es wird dann eine neue *ScatterplotMatrix* instantiiert. Wurder vorher bereits eine *ScatterplotMatrix* erzeugt, wird diese gelöscht. Es ist somit zu jeder Zeit immer nur eine *ScatterplotMatrix* in der Szene. Über das Attribut

pointSize kann die Größe der *DataPoints* angepasst werden. Der *ImportDialog* inkrementiert bzw. dekrementiert diesen Wert, wenn der Benutzer die entsprechenden Buttons klickt.

Programmablauf

Das folgende Diagramm stellt den Programmablauf beim erstmaligen Starten der Anwendung dar:

- Blub
  - Blub
- blub

Blub

1. Ich
   a. Mag
      i. Züge
2. du
   a. magst
      i. LKW
         A. Tut Tut

Neue Subüberschrift

Blub Mein Bereich

Beispiel 1. Mein Bereich

```
Übernimmt Ascii Zeug
```

Listing 2. Source Code

```
public void Foo(){

}
```

Und noch Ref auf Überschrift Neue Subüberschrift.

Quellen / Ressourcen

- [MiddleVR]: https://www.middlevr.com/home
- [IATK] Immersive Analytics Toolkit: https://github.com/MaximeCordeil/IATK
- [VRTK] Virtual Reality Toolkit: https://vrtoolkit.readme.io
- [SteamVR]: https://steamcommunity.com/steamvr

- [GitHub-Repository] Projekt Repository: https://github.com/MHaselmaier/VR-Visualization.git
- [Unity-Documentation]: https://docs.unity3d.com/Manual/index.html
- [MiddleVR-Documentation]: https://www.middlevr.com/doc/current/