



Universitätsname
Fachbereich Name

Logo

Titel

UNTERTITEL

Seminar: Seminarname
Betreuer: Betreuer 1
Betreuer 2

Teilnehmer: Teilnehmer 1,
Teilnehmer 2,
Teilnehmer 3,
Teilnehmer 4,
Teilnehmer 5

Inhaltsverzeichnis

| | |
|---|-----------|
| Kapitel 1: Einleitung | 5 |
| 1.1 Weitere Unterkapitel und das Inhaltsverzeichnis | 5 |
| Kapitel verschieben | 6 |
| Weitere Kapitelebenen | 6 |
| 1.2 Generelles über Formatvorlagen | 6 |
| 1.3 Einfügen von Grafiken | 6 |
| Kapitel 2: Weiter geht's | 7 |
| 2.1 Literaturverzeichnis | 7 |
| 2.2 Hinweise innerhalb der Hausarbeit | 7 |
| Kapitel 3: Beispieltext | 9 |
| 3.1 Beschreibung der .m-Files | 9 |
| 3.2 Beschreibung der Datenstrukturen | 10 |
| Namenskonventionen | 11 |
| Wichtige Variablen | 12 |
| Kapitel 4: Literaturverzeichnis | 15 |

1 Einleitung

Die obligatorische Einleitung der Hausarbeit. Beachte, dass die Vorlage auf automatische Kapitelnummerierung und automatische Erstellung von Inhaltsverzeichnissen ausgelegt ist. Das heisst, um ein neues Kapitel zu beginnen, gehe in die nächste Zeile, drücke F11 und wähle dann die Formatvorlage „Überschrift 1“ aus, und schreibe die Überschrift. Durch ein Return springt man nicht nur in die nächste Zeile. Es wird automatisch zur Formatvorlage „Textkörper“ gewechselt, so dass man direkt im richtigen Layout weiter schreiben kann. Für Absatz-Zitate gilt natürlich das gleiche:

„Einfach in die nächste Zeile gehen, die Formatvorlage „Zitat“ auswählen und los schreiben. Auch hier gelangt man durch ein Return wieder zur Textkörpervorlage. Soll also ein Absatzzitat über mehrere Absätze gehen, muss die Formatvorlage neu ausgewählt werden.“¹

Die gesamte Vorlage ist auf das Verwenden von Formatvorlagen ausgelegt. Ein Unterkapitel der Ebene 2 besitzt die Formatvorlage „Überschrift 2“

1.1 Weitere Unterkapitel und das Inhaltsverzeichnis

Auch hier findet selbstverständlich eine automatische Nummerierung statt. Möchte man nach geschriebenem Text das Inhaltsverzeichnis aktualisieren, genügt ein Rechtsklick auf das Inhaltsverzeichnis (der Cursor muss sich allerdings im Inhaltsverzeichnis befinden) und ein Klick auf „Aktualisieren“.

¹ Mit Strg-Shift-F kann man automatisch eine Fusszeile setzen. Mit Bild-Rauf kommt man zur entsprechenden Textstelle wieder zurück

Kapitel verschieben

Die dritte Kapitelebene kann über die Formatvorlage „Überschrift 3“ erreicht werden. Kapitel können auch nachträglich einfach verschoben und in ihrer Hierarchie verändert werden. Dies lässt sich am einfachsten über den Navigator erreichen, der über F5 geöffnet kann. Hier können einzelnen Kapitel angeklickt und über entsprechende Buttons verändert werden. Das Menü ist weitgehend selbsterklärend.

Weitere Kapitelebenen

Weitere Kapitelebenen sind in dieser Dokumentvorlage nicht enthalten, können aber problemlos eingerichtet werden. Über „Extras“ - „Kapitelnummerierung“ können für die tieferen Ebenen weitere Vorlagen (sinnvollerweise „Überschrift 4“ usw.) ausgewählt werden und für diese eine Zählart festgelegt werden. Beachte dabei, dass Ebene 10 bereits für die Formatvorlage „Überschrift“ vorgesehen ist, die dazu verwendet wird, Kapitel, wie „Inhaltsverzeichnis“, „Abbildungsverzeichnis“, „Symbolverzeichnisse“ ohne Nummerierung zu gestalten, in der Inhaltsverzeichnis-hierarchie aber trotzdem ganz links erscheinen zu lassen. Letztendlich stehen also eigentlich nur die Ebenen 1-9 zur Verfügung.

1.2 Generelles über Formatvorlagen

Versuche, ausschliesslich Formatvorlagen beim Arbeiten zu verwenden. Das hat den Vorteil, dass Änderungen einheitlich vorgenommen werden und inkonsistente Formatierungen viel leichter vermieden werden können. Neben den gewöhnlichen Absatzvorlagen gibt es auch Zeichenvorlagen, wie „betont“ oder „stark betont“. Auch „*zeichenweises Zitieren sollte layouttechnisch durch eine Formatvorlage*“ umgesetzt werden. Die normale Formatvorlage lautet dabei „Standard“.

1.3 Einfügen von Grafiken

Damit auch das Abbildungsverzeichnis sinnvoll ausgestaltet werden kann, macht es Sinn, zu jedem Bild eine Beschriftung hinzuzufügen. Bilder werden idealerweise wie folgt eingefügt: In eine neue Zeile gehen, Formatvorlage „Abb.“ auswählen, Bild einfügen, Rechtsklick auf's Bild und „Verankerung“ - „Als Zeichen“ auswählen (Bild erscheint nun zentriert). Rechtsklick auf's Bild, „Beschriftung...“ anklicken, Leerzeichen und die eigentliche Beschreibung angeben, Kategorie „Abb.“ auswählen und bestätigen.

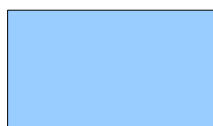


Abb. 1 Das ist ein Rechteck

Aus mir unbekanntem Gründen lässt sich eine Beschriftung nachträglich nicht mehr auf diese Weise ändern. Änderungen am Text können darum nur direkt in der Abbildung vorgenommen werden.

2 Weiter geht's

Beachte, dass mit einer Kapitelüberschrift der Ebene 1 auch immer eine neue rechte (!) Seite beginnt. Das bedeutet: wenn der Text im vorhergehenden Kapitel auf einer rechten Seite endet, wird automatisch eine leere linke Seite eingefügt. Diese leere Seite ist im Bearbeitungsmodus nicht sichtbar. Klickt man jedoch auf „Datei“ - „Seitenansicht“, so lässt sich jedoch die genaue Anzahl der Seiten und die eventuell leer eingefügten Seiten einsehen. Ist diese Funktion eher nicht gewünscht, kann diese leicht abgeschaltet werden. Rechtsklick auf die Formatvorlage „Überschrift 1“ - „Ändern“ - „Textfluss“ - „Umbrüche einfügen“ deaktivieren.

Einstellungen an den Seitenrändern können in den Seitenvorlagen vorgenommen werden. Dabei kann das Layout der ersten Seite („Erste Seite“), der Verzeichnisseiten („Verzeichnis“) und der normalen Hausarbeitsseiten („Standard“) unabhängig voneinander geändert werden. Über „Einfügen“ - „Manueller Umbruch“ kann problemlos ein Wechsel zwischen zwei Seitenformatvorlagen vorgenommen werden.

2.1 Literaturverzeichnis

Diese Formatvorlage unterstützt auch das automatische Erstellen eines Literaturverzeichnisses. Unter „Extras“ kann dazu die Literaturdatenbank aufgerufen werden und mit der verwendeten Literatur gefüllt werden. Möchte man im Text zu einem Zitat oder irgendwelchen Behauptungen eine Literaturangabe vornehmen, geht man auf „Einfügen“ - „Verzeichnisse“ - „Literaturverzeichniseintrag..“. Dort kann man dann das entsprechende Kürzel auswählen. Verwendete Literaturhinweise werden dann in aller Ausführlichkeit im Literaturverzeichnis aufgeführt.

2.2 Hinweise innerhalb der Hausarbeit

Möchte man innerhalb der Hausarbeit auf andere Kapitel verweisen, so ist es ratsam, auch dies dynamisch zu regeln. Angenommen, ich möchte von hier aus auf das Kapitel „Kapitel verschieben“ verweisen. Zunächst geht man dazu auf die Kapitelüberschrift, markiere diese und

geht auf „Einfügen“ - „Textmarke...“. Dort gibt man einen Namen ein und bestätigt. Wenn man nun auf 1.1 Kapitel verschieben auf Seite 6 verweisen möchte, geht man auf „Einfügen“ - „Feldbefehl“ - „Andere“ - „Referenzen“ - Feldtyp: Textmarken, wählt dort seine Textmarke aus und fügt alle Informationen, die man braucht, in den Text ein. Verschiebt sich die Textmarke irgendwann auf eine andere Seite oder bekommt sie eine andere Kapitelnummer, so ändert sich dies auch automatisch in den eingefügten Feldern.

3 Beispieltext

Hier noch ein wenig Beispieltext, damit man mal die Vorlage in Aktion begutachten kann. Dieses Kapitel wendet sich an alle Programmierer und Entwickler, die die internen Strukturen von Icynator verstehen wollen, um das Programm modifizieren oder erweitern zu können. Da die meisten Funktionen durch Callbacks aufgerufen werden, die sich leicht verfolgen lassen und der Code ausführlich kommentiert ist, ist für den schnellen Einstieg in einzelne Programmteile lediglich ein grobes Verständnis des gesamten Programmes notwendig. Im Folgenden werden daher die Funktionen aller für das Programm wichtigen .m-Dateien erklärt, sowie Informationen über Namenskonventionen, Variablenstruktur etc. gegeben.

3.1 Beschreibung der .m-Files

Im folgenden werden alle für das Programm relevanten .m-Dateien erklärt. ICA-Algorithmen, die ebenfalls in .m-Dateien ausgelagert sind, sind intern nicht mit Icynator verbunden, weshalb deren Beschreibung den Kommentaren des Codes beiliegt.

| Datei | Beschreibung |
|------------------|---|
| addAlgo.m | Öffnet ein Fenster, in dem der Benutzer die Algorithmen auswählen kann, die er testen möchte. Klickt der Benutzer auf „Öffnen“ werden die ausgewählten Algorithmen der Listbox hinzugefügt. Die Rückgabeveriable <i>succ</i> gibt an, ob der Benutzer erfolgreich Algorithmen selektiert oder diesen Prozess abgebrochen hat. |
| addData.m | Öffnet ein Fenster, in dem der Benutzer die Signaldaten auswählen kann, die entmischt werden sollen. Wie bei <i>addAlgo.m</i> kümmert sich die Unit darum, die Daten zu laden und den Erfolg dieses Prozederes an die Hauptunit <i>benchmarking.m</i> zurück zu melden. |

| Datei | Beschreibung |
|---------------------------------|--|
| addNoise.m | Fügt Rauschen zu den Signaldaten hinzu. Als Eingabe werden die Signaldaten und die Rauschdaten übergeben. Zurückgeliefert werden die vermischten Daten. Die Generierung der Rauschdaten bezüglich der Signaldaten (Skalierung, Stärke, etc.) geschieht mit den im Ordner <i>noiseterms</i> enthaltenen Algorithmen. |
| benchmarking.m | Das Hauptprogramm. Die Unit, die alle Callback-Funktionen der Hauptanwendung, sowie die nötigen Abläufe für das ICA-Verfahren enthält. Von hier aus wird auf die Funktionen der anderen .m-Files zugegriffen. |
| createParametersVector.m | Erstellt ein Cell-Array mit den Parametern, mit denen ein Algorithmus aufgerufen werden soll. Der Benutzer hat die Möglichkeit im Parameter-Assistenten festzulegen, mit welchen Parametern der Algorithms aufgerufen wird. Da er hier auch Matlab-Ausdrücke, wie Vektoren oder Schleifen angeben kann, müssen diese im Vorfeld ausgewertet werden. Dies kann dazu führen, dass durch die Eingabe der Parametereinstellungen ein Algorithmus nicht einmal aufgerufen wird, sondern entsprechend der Anzahl der angegebenen Vektorelemente. Jede Zelle des von dieser Unit zurückgegebenen Arrays enthält damit die Parameter für einen Algorithmusauf-ruf. |
| dbAdress.m | Das Abfragefenster für die Daten, die für eine Verbindung mit einer Datenbank notwendig sind. Diese Daten werden in der Datei <i>dbInfo.mat</i> abgespeichert. |
| dbClearAlive.m | Setzt das alive Attribut für alle Einträge in der aktuellen Tabelle, die den Wert „1“ oder „2“ hatten, auf „0“. Damit sind alle Einträge permanent abgespeichert. |
| dbClearTempData.m | Löscht alle Einträge in der aktuellen Tabelle, bei denen das alive Attribut den Wert „2“ hat. |
| dbConnect.m | Verbindung mit der Datenbank wird hergestellt. Dies geschieht anhand der in <i>dbInfo.mat</i> enthaltenen Datenbankin-formationen. |
| dbCreateTable.m | Erstellt eine Tabelle in der gewählten Datenbank. Als Parameter wird der Namen der zu erstellenden Tabelle übergeben. |

3.2 Beschreibung der Datenstrukturen

Die wichtigste und vielseitigste Datenstruktur in Matlab ist die Klasse *struct*, mit der zum Beispiel die Steuerung der GUI funktioniert und über die große Datenmengen leicht gehandhabt werden können. Auch im Icynator wird diese Klasse eingesetzt, um Daten zu den einzelnen Modulen zu übertragen. Damit das Programm leicht modifiziert werden kann, hat sich die Projektseminar-gruppe auf Namenskonventionen geeinigt und Datenstrukturen entworfen, die überall im Pro-gramm in der gleichen Weise verwendet werden. Die folgenden Unterkapitel dokumentieren die-se Konventionen.

Namenskonventionen

In Icynator gibt es eine Reihe von Präfixe, die zur Kennzeichnung bestimmter Variablengruppen benutzt werden. Diese Präfixe sollten auch bei eigenen Erweiterungen unbedingt eingehalten werden, da sie das Debuggen und Reviewen des Codes erheblich erleichtern.

| Präfix | Bedeutung |
|---------------|--|
| btn | GUI-Button |
| lbox | GUI-Listbox |
| pan | GUI-Panel |
| cb | GUI-Checkbox |
| popmenu | GUI-Popupmenu |
| edit | GUI-Eingabefeld |
| txt | GUI-Textlabel |
| axe | GUI-Plotterpanel |
| men | GUI-Menü |
| b | Boolische Variable |
| index_ | Index-Variable für die Speicherung selektierter Einträge |
| noise_ | Variable für Rausch-Informationen |
| loop_ | Variable für Loop-Informationen |
| db | Datenbankvariable |
| Load | Funktion zum Laden von Daten |
| Save | Funktion zum Speichern von Daten |

Für alle GUI-Elemente legt Matlab automatisch Callback-Funktionen an und übernimmt dabei die verwendeten Elementnamen. Eine Callback-Funktion hat die Form

$$\langle \text{GUI-Name} \rangle_Callback$$

Dementsprechend haben praktisch alle relevanten Funktionen ebenfalls ein Präfix nach den oben aufgeführten Namenskonventionen. Ein Button namens *btnAddAlgo* hat damit zum Beispiel die Callback-Funktion *btnAddAlgo_Callback*. Alle Variablen, die keines der oben aufgeführten Präfixe besitzt, sind Variablen vom Typ *double*, da Matlab unterschiedliche Zahlentypen, wie Integer, Double, Cardinal etc. nicht unterscheidet. Strings werden innerhalb von Matlab als Char-Ketten gesehen, was dem Typ *array* entspricht.

Wichtige Variablen

Um innerhalb der verschiedenen Callback-Funktionen der Hauptanwendung auf alle relevanten Daten zugreifen zu können, führten wir eine globale Struktur namens *ICA* ein. Diese Struktur beinhaltet alle eingestellten Algorithmen, sowie die zu entmischenden Rohdaten samt Mischmatrizen. Die Struktur besitzt folgende Variablen und Untervariablen:



Abbildung 3.1 Die Variablenstruktur

Möchte man zum Beispiel auf den Dateinamen des zweiten in die Anwendung geladenen Algorithmus zugreifen, so lautet der Befehl dafür

```
> ICA.algo(2).filename
```

Neben den drei Arrays speichert die ICA-Struktur noch den Index des selektierten Algorithmus und der selektierten Sounddaten, um diese nicht jedesmal neu aus den GUI-Objekten lesen zu müssen. Das Array `result` wird nur angelegt, sobald der Benutzer auf den Run-Button klickt, wodurch die Callbackfunktion `btnRun_Callback` gestartet wird.

Da jedem Testdatensatz beliebig viele Matrizen zugeordnet werden können, besitzt ein `data`-Eintrag ein Array mit allen ihm zugewiesenen Matrizen. Auf den Namen der dritten Matrix des zweiten Testdatensatzes kann man demzufolge über folgenden Befehl zugreifen.

```
> ICA.data(2).matrices(3).name
```

Der gesamte Testlauf wird also konfiguriert, in dem in den Callback-Funktionen der jeweiligen GUI-Elemente die Arrays mit den gewünschten Algorithmen und Testdaten samt Matrizen konstruiert werden. Wird der Testlauf gestartet, wird einfach durch alle Array-Einträge iteriert.

4 Literaturverzeichnis

- Andrzej Cichocki, Shun-ichi Amari: Adaptive Blind Signal and Image Processing
- Aapo Hyvärinen, Juha Karhunen, Erkki Oja: Independent Component Analysis
- Mark Girolami: Self-Organising Neural Networks for ICA and BSS
- Beate Meffert, Olaf Hochmuth: Werkzeuge der Signalverarbeitung
- Daniel Schobben, Kari Torkkola, Paris Smaragdis: Evaluation of Blind Signal Separation Methods
- Ali Mansour, Mitsuru Kwamoto, Noboru Ohnishi: A Survey of The Performance Indexes of ICA Algorithms
- Russell H. Lambert: Difficulty Measures and Figures of Merit for Source Separation
- Jerome H. Friedman: Exploratory Projection Pursuit
- Xavier Giannakopoulos: Comparison of Adaptive Independent Component Analysis Algorithms
- Remi Gribonval, Laurent Benaroya, Emmanuel Vincent, Cedric Fevotte: Proposals for Performance Measurement in Source Separation