Elo Trainingsmodule

Onderzoeksverslag

TUDelft | NSDSK

Inleiding

In dit onderzoeksverslag zullen we laten zien welke keuzes we in het ontwikkelproces hebben moeten maken. Een aantal keuzes konden we vooraf maken en hebben we bepaald in het vooronderzoek. Tijdens het project echter zijn we op een aantal problemen gestuit waardoor we de plannen toch moesten aanpassen. We laten daarom zien welke keuzes we konden behouden en welke we moesten aanpassen.

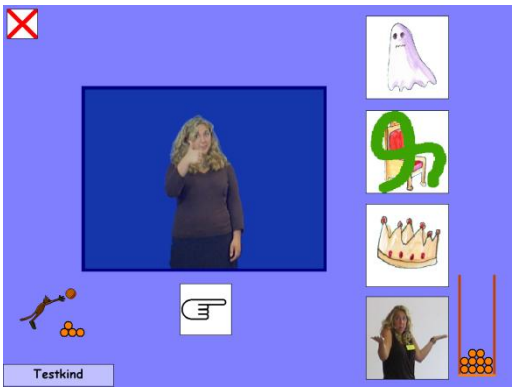
ELO

In een samenwerkingsverband tussen de NSDSK, de TU-Delft en de Koninklijke Auris Groep Rotterdam wordt gewerkt aan het ontwikkelen en beproeven van een elektronische leeromgeving (ELo) voor het leren van actieve en passieve Nederlandse Gebarentaal vocabulaire aan jonge kinderen met ernstige auditieve beperkingen.
Elo bevat vier oefeningen:

1. de vertelplaatoefening:
   Het beeldscherm toont een vertelplaat en het kind krijgt door bepaalde onderdelen van de vertelplaat aan te raken het bijbehorende gebaar in een videoclip te zien

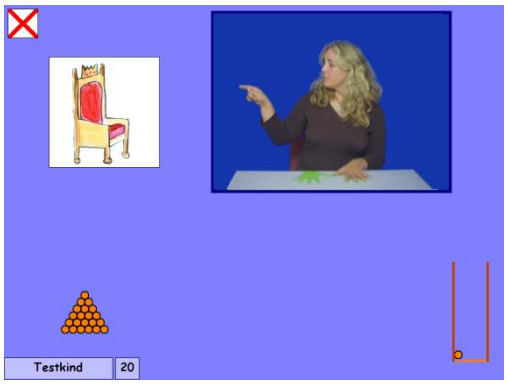
Figuur 1

2. de begripsoefening:
   Een video speelt een gebaar af en toont vervolgens een aantal plaatjes. Het kind wordt gevraagd het plaatje aan te wijzen dat bij het gebaar hoort. Daarna krijgt het kind feedback. De feedback wordt getoond aan de hand van een videoclip, waarbij in gebarentaal wordt aangegeven dat het antwoord goed is (er verschijnt een groene krul door de antwoordkeuze) of fout (er verschijnt een rood kruis door de antwoordkeuze). Bij een foutief antwoord wordt tevens het correcte antwoord getoond.



Figuur 2

3. de productieoefening:
   Het kind ziet een plaatje en wordt uitgenodigd het gebaar dat bij het plaatje hoort te maken. Hierna geeft ELo direct feedback in termen van goed of fout. In geval van een incorrect gebaar toont ELo, evenals bij de begripsoefening, het correcte gebaar.



Figuur 3

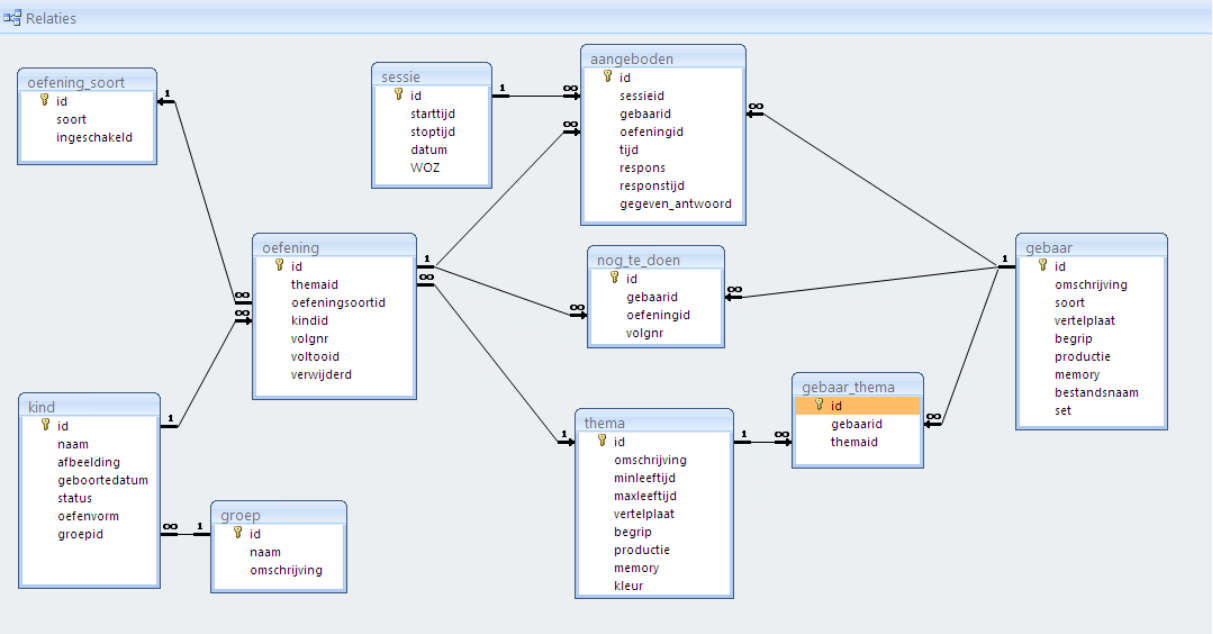
4. De memorie oefening:
   Het kind kiest steeds een plaatje bij een gebaar. Hierna geeft ELo direct feedback in termen van goed of fout.

Figuur 4

De vraag is wat er precies allemaal nodig is om een gebaar aan het elo systeem toe te voegen. Voor oefening 1 tot en met 4 zijn onderstaande elementen nodig:

1. Plaatje van het gebaar
2. Voorbeeld filmpje van het gebaar
3. Een detector om het gebaar via de camera te detecteren
4. Een vertelplaat waarop figuren kunnen worden aangeklikt

Al deze gebaren worden momenteel opgeslagen in een database, de onderlinge relaties van tabellen in deze database wordt hieronder afgebeeld.



Figuur 5

Als we dus een gebaar aan elo toe willen voegen moet dit gebaar in de tabel ‘gebaar’ gezet worden. Verder moet een gebaar lid kunnen worden gemaakt van een één of meerdere thema’s.

Voorts kwamen we er tijdens deze onderzoeksperiode achter dat ELO nog wel eens vastloopt als men zich niet in de ideale omgevingen bevindt.

Elo is opgebouwd uit 3 verschillende talen:

- De interface is gemaakt in Visual basic 6.0
- Het rekenwerk zoals de herkenning wordt gerealiseerd met behulp van matlab
- Onder andere het besturen van de camera’s gebeurt vanuit C++

Bij ons kwam direct de vraag naar boven waarom men zich niet beperkt had tot één taal in plaats van drie. Na enig onderzoek kwamen we tot de volgende conclusies:

- De interface is in visual basic geschreven omdat de interface tool van matlab niet alle mogelijkheden bevat die nodig zijn voor het elo programma.
- Het rekenwerk wordt in matlab uitgevoerd omdat dit de ideale omgeving is voor bijvoorbeeld het rekenen met matrices.

- De rede dat de camera's vanuit c++ worden bestuurd komt voort uit het feit dat matlab hiervoor niet afdoende is.

Middelen

We wilden de trainingsmodule goed laten aansluiten in Elo. We hadden een demonstratie van Elo gehad bij het NSDSK. In Elo kun je door in het menu op 'T' te klikken in de 'Teachers module' komen. Ons leek het handig om ook door middel van een toets in de trainingsmodule te komen. De interface van Elo was gebouwd met Visual Basic 6 in Visual Studio 6. De ondersteuning van Visual Studio is echter dit jaar komen te vervallen [1]. Daarom was het voor ons geen optie verder te gaan met Visual Basic 6.
Nieuwere versies van Visual Studio bieden de mogelijkheid om projecten die gebouwd zijn met een oudere versie om te zetten naar de nieuwe versie. Zo ook van het oude Visual Basic naar het .Net Framework. Talen waar we de afgelopen tijd intensief mee hebben gewerkt zijn java en C-sharp. De vraag was nu of we ons project ook in Visual Basic moesten schrijven of dat we toch beter konden kiezen voor een wat bekendere taal. Uit een aantal artikelen op internet bleek dat Visual basic .NET grote overeenkomster vertoont met C# [2], dit zou dus zeker in ons voordeel werken.

Doordat we één van de vorige projecten in C-sharp .Net hebben geschreven zijn we dus bekend met het .Net Framework. Het gebruiken van Visual Basic .Net was voor ons dan ook een goede optie.
We hadden een eventueel alternatief in Matlab. Matlab functies werden in het Elo project al gebruikt, en de hele training was gebouwd in Matlab. Uit ervaring met een eerder project wisten wij dat het bouwen van een Gui met meerdere schermen niet handig en overzichtelijk was. Ook in .Net hadden we ervaring in het maken van een Gui en dit was een stuk gemakkelijker dan Matlab.
Dit alles in oogschouw genomen hebben we daarom gekozen voor Visual Basic .Net als basis voor dit project.

Het zou dus mooi zijn als het huidige project omgezet kon worden naar Visual Basic .Net. Na enig onderzoek kwamen we er achter dat we het project automatisch konden migreren of dat elo beter opnieuw opgebouwd kon worden [3]. Er werden wel enkele nadelen aangegeven van de automatische migratie, zoals het niet optimaal gebruik maken van de mogelijkheden van .Net en het feit dat alle programmatuur uitgebreid opnieuw moet worden getest. Ondanks deze nadelen hebben we toch geprobeerd om met behulp van Visual Studio .Net het project om te zetten naar .Net. We zijn hier de eerste 2 dagen mee bezig geweest. Er waren echter teveel functies die in .Net niet meer beschikbaar waren. Het ging dan ook teveel tijd kosten om Elo werkend te krijgen in .Net terwijl dat ons doel niet is. We zijn daarom een nieuw Visual Basic .Net project begonnen. Als het Elo project later verder opgepakt wordt en het omgezet wordt naar .Net kan de trainingsmodule daarop aangesloten worden. Het is echter wel aan te raden om hierbij met een nieuw project te starten en geen, of beperkt, gebruik te maken van de automatische migratie tools.

We wilden voor de Matlab functies gaan werken met de op dit moment gangbare Matlab versie. Dit was de versie uit 2007. Het bleek echter zo te zijn dat als functies met Matlab gecompileerd werden, je ook de bijbehorende compiler nodig had [4]. Een hogere of lagere versie is niet te gebruiken. We hebben toen besloten de nieuwste versie te gaan gebruiken (2008a). De kans is daarmee groter dat bij het verder werken aan dit project ook die versie gebruikt gaat worden. Dan hoeven niet alle projecten opnieuw gecompileerd te worden. Het is mogelijk om meerdere compilers te installeren,

maar dit is uiteraard niet wenselijk. De keuze om Matlab te blijven gebruiken is eigenlijk triviaal, omdat alle functies uit de training in Matlab geschreven zijn. Het zou een enorm karwei worden om alles om te zetten naar een eventuele andere taal. Het moest uiteraard wel mogelijk zijn de Matlab functies te importeren in .Net. Matlab zelf bevat een COM Builder en na enig uitleg gezocht te hebben, konden we hiermee een dll maken die geïmporteerd werd in Visual Basic [5]. Daar deze COM Builder ingebouwd was in Matlab hebben wij ook niet gekeken naar alternatieven. Tijdens het doorlezen van deze uitleg kwamen we erachter dat er nog een extra dll geïmporteerd moest worden. De zogenaamde MWArray.dll, deze is nodig om alle objecten van Visual Basic om te zetten naar de objecten van Matlab. Als wij deze dll niet gebruikten kregen we te maken met compatibiliteitsproblemen tussen de verschillende objecttypes.

Het exporteren was een punt waar enig onderzoek naar gedaan moest worden. Het huidige ELO programma werd geïnstalleerd door een autorun bestand die in een dos scherm alle nodige bestanden installeerden. Dit is niet erg gebruikersvriendelijk en komt ook niet echt professioneel over. Dit was dan ook de reden dat wij op zoek gingen naar een betere manier. Op internet waren redelijk wat tools te vinden om een automatische installer te maken. Het nadeel van deze tools was dat ze redelijk uitgebreid waren en dat je er vaak voor moest betalen. Uiteindelijk bleek Visual Studio 2008 een goede setup wizard te bevatten [6], we hebben dan ook direct voor deze optie gekozen.

Uiteraard is er ook een database nodig om de voortgang van de training bij te houden. Met het opslaan van gebaren, opnames, nieuwe thema's etc.
De database moest hoe dan ook aansluiten op de database van Elo. Bij de training kan de gebruiker namelijk aangeven welk gebaar gekoppeld moet worden aan welk thema. Nieuwe thema's moeten toegevoegd en verwijderd kunnen worden. Daarvoor is in ieder geval de tabel thema nodig uit de oorspronkelijke Elo database. Vanwege deze integratie met het oorspronkelijke Elo systeem hebben wij besloten dezelfde database te gebruiken. Dit is een Microsoft Access Database. De standaard driver die Microsoft heeft om verbinding te maken met de Access Database is de ADO database driver [7]. Deze driver werd ook al gebruikt in het Elo systeem. We hebben dan ook niet gekeken naar alternatieven voor de driver. Wel hebben we onderzocht hoe de gebruiker deze driver op de installatie pc kon krijgen. Uit wat documentatie bleek namelijk dat deze driver niet standaard in windows zit, maar dat er wel een package is waar deze driver in zit [8]. Dit zou echter betekenen dat er wederom een extra handeling zou moeten worden verricht door de gebruiker. Vandaar dat we vervolgens naar de mogelijkheid hebben gekeken om deze direct in ons project te plaatsen [9]. Dit bleek vrij eenvoudig, vandaar dat we uiteindelijk ook voor deze optie hebben gekozen.

Video's

Het opnemen van video's werd in de oude situatie gedaan met Fvfm. Dit programma was gemaakt in c++ en gecompileerd werd dit vanuit Matlab gebruikt. Wij wilden zoveel mogelijk naar Visual Basic halen zodat we niet teveel externe functies nodig hadden en naar ons idee moest het ook kunnen om vanuit Visual Basic video's op te nemen. Langzaam ging dit ook lukken, we konden de drivers van de camera aansturen met .Net componenten die meegeleverd waren in Firepackage. De meegeleverde documentatie van Firepackage [10] bood ons voldoende inzicht in de werking van de componenten.
De beelden van de 2 camera's konden we in een Picturebox plaatsen, de camera kon gestopt en gestart worden. Het was alleen wel noodzakelijk een minimale framerate van 30 te halen. Met

slechts het draaien van de camera konden we dit met gemak halen. Echter de videobeelden moeten ook opgenomen worden, we kwamen hierbij niet hoger dan een framerate van 20.

De buffer van Visual Basic bleek niet goed genoeg om de stroom aan beelden te verwerken. In de documentatie van de .Net componenten konden we niet vinden waar dit eventueel aan zou kunnen liggen. Na verder gezocht te hebben op internet konden we nergens vinden waar we dit probleem eventueel konden verhelpen. We hebben gezocht naar een andere oplossing door de Firepackage componenten niet te gebruiken en drivers te zoeken die de camera's konden aansturen, we konden echter geen drivers hiervoor vinden.
We waren daarom ook genoodzaakt van dit plan af te stappen en toch terug te keren naar Fvfm. Fvfm moest echter wel aangestuurd worden vanuit Visual Basic, omdat dit onze basis was waar vanuit we werkten. In Elo werkt de herkenner met gecompileerde Matlab bestanden en aangezien Fvfm ook vanuit Matlab aangestuurd kon worden, hebben we een Matlab project gestart. Met Matlab functies konden we Fvfm gebruiken. Fvfm hadden we geïmporteerd in het Matlab project. Dit totaal is omgezet naar een .Net component welke weer te gebruiken was in Visual Basic. Met het .Net component kunnen de functies van Matlab aangestuurd worden. De cameraschermpjes hebben we in Fvfm zelf op een goede plaats gezet, zodat ze op een goede manier in het menuscherm vielen.

Het afspelen van de video's wilden we met de windows media player component van .Net doen. Echter Fvfm maakt bij het opnemen van video's een .raw en een .mat bestand aan. Deze kunnen omgezet worden naar een .avi bestand die de media player kan afspelen. Na elke keer opnieuw opstarten moet dan wel worden aangegeven welke codecs gebruikt gaan worden en welke compressie je wilt hebben. Het leek ons niet wenselijk voor de gebruiker om dit steeds aan te geven. We hebben er daarom voor gekozen om het afspelen van video's ook door Fvfm te laten doen. De schermpjes staan immers al op de goede plaats voor het opnemen. En omdat we Fvfm in ieder geval voor het opnemen moeten gebruiken konden we het afspelen ook prima door Fvfm laten gebeuren. We hebben hier alleen een tweetal extra functies in Matlab voor nodig gehad. Een voor het starten van de video en een voor het stoppen ervan.

Buiten dit alles hadden we nog een ander probleem waar we tegenaan liepen. Elo maakt gebruik van een bepaalde Firepackage. Deze versie had echter nog geen .Net componenten beschikbaar. In de nieuwste versie waren deze wel beschikbaar, maar daarvoor moest de oude versie vervangen worden. Elo kon helaas niet overweg met de nieuwe firepackage. Elo heeft een module prototype (gemaakt met behulp van Matlab), prototype bevat tevens de bestanden van de oude firepackage. In feite moet er een nieuwe prototype gemaakt worden met de bestanden van de nieuwe firepackage om het werkend te krijgen. Hier zijn we niet mee verder gegaan, omdat het veel tijd ging kosten om uit te zoeken wat daar allemaal bij moet. Bovendien zouden diverse onderdelen van Elo ook aangepast moeten worden. Uiteindelijk hebben we ervoor gekozen toch Fvfm te gebruiken wat eerder vermeld is.

Training

Voor het project begon hadden we alleen de bestanden van het Elo programma en niets van de training zoals die op de oude wijze doorlopen werd. Van de training kregen we vervolgens een gui die in Matlab geopend kon worden. In de handleiding stond een klein stukje over de stappen die in de training doorlopen moesten worden. Hierbij werd helaas niet uitgelegd welke bestanden daar

precies voor nodig waren. De gui gaf ons hierbij ook niet veel meer duidelijkheid. Uit de handleiding konden we halen dat er 4 stappen genomen moesten worden; een stap extract features, makedataset, een warping stap en de uiteindelijke training zelf. Alle bestanden voor de training stonden in 1 map, maar er stond niet bij welk bestand voor welke stap nodig was. Uiteindelijk bleek dat we ook niet alle bestanden hadden die nodig waren. Na een paar weken hadden we alles verzameld en konden we in principe de training doorlopen. Toch konden we niet goed achterhalen wat er precies gebeurde. We kregen gelukkig hulp van de maker van deze software. Hij had voor ons de gui wat overzichtelijker gemaakt met de knoppen voor de diverse stappen achter elkaar. Uit de methodes achter deze knoppen konden we bepalen welke functie werd aangeroepen.

Omdat wij alles zo modulair mogelijk aan het opbouwen waren, wilden we de stappen ook gescheiden kunnen uitvoeren. Bij het exporteren van een *.m file met behulp van de 'deploytool' van matlab worden automatisch alle bestanden die de m file gebruikt bijgevoegd [11]. Het probleem is echter dat matlab niet verder kan kijken als hij een dll tegen komt. Hierdoor voegde de compiler niet alle afhangelijke bestanden toe en waren we genoodzaakt om een trial-and-error methode toe te passen. De functie die de stap als het ware inluidt, hadden we in een aparte map gezet en toen aangeroepen vanuit Matlab. In Matlab konden we aan de foutmeldingen precies zien welke bestanden daadwerkelijk nodig waren. Op deze manier hadden we uiteindelijk voor elke stap een aparte map met functies. Deze stappen moesten uiteraard naar een .Net component omgezet worden om in Visual Basic te gebruiken. Alle functies moesten we dan ook goed doorlopen om te bepalen wat de input precies moest zijn. Daarbij kwamen we erachter dat veel variabelen niet meer variabel waren, maar hard-coded in het programma waren gezet. Toen we alles hadden klaar gemaakt voor een test in Visual Basic bleek het nog niet goed te werken. Dit kwam doordat in 1 functie 'sign2number' alle gebaren uit elo in een array opgeslagen stonden zodat die omgezet konden worden naar een nummer. Het bleek dus nodig om een array te hebben van alle gebaren die getraind moeten worden. Dit hebben we opgelost door aan het begin van de training een .mat bestand te maken waarin alle gebaren zijn opgeslagen. Het pad naar dit bestand werd meegegeven aan de functies zodat daaruit alle benodigde informatie gehaald kon worden. Op deze manier konden we de training in zijn geheel vanuit Visual Basic doorlopen.

Aan de start van dit project hadden we weinig theorie over de training. Er stond een klein stukje over de trainingsstappen in de handleiding [12], maar daar hadden we in feite niet veel aan. We hebben de hele code doorlopen om te begrijpen wat er precies gebeurde. Kleine delen code hebben wij tijdens het proces ook verwijderd, omdat dit vaak onnodig was. We hebben een paper [13] gevonden waarin beschreven stond hoe de warping (Dynamic Time Warping, DTW) en de CDFD training (Combined Discriminative Feature Detectors) werkten. Met behulp van deze paper en de code konden we uiteindelijk nagaan hoe een en ander in zijn werk ging. Dit was de enige paper waarin de CDFD training vermeld stond. De paper was dan ook onder andere geschreven door de maker van de oorspronkelijke Elo software.

Referenties

[1] http://msdn.microsoft.com/en-us/vbrun/ms788707.aspx, einde support visual basic 6.0
[2] Visual Basic .Net versus c# – Remi Caron, .Net magazine for developers #7, 2004
[3] Overstappen van Visual Basic 6.0 naar Visual Basic .Net – Gert-Jan Messchendorp, .Net magazine for developers #7, 2004

[4] http://www.mathworks.com/support/solutions/data/1-1IW46N.html?product=CO&solution=1-1IW46N, Omschrijving van het versie verschil probleem
[5] http://www.mathworks.com/products/netbuilder/, Werking COM Builder van Matlab
[6] http://msdn.microsoft.com/en-us/library/kay7az51.aspx, uitleg Setup Wizard in Visual Studio 2008
[7] http://msdn.microsoft.com/en-us/library/ms675532(VS.85).aspx, adodb driver
[8] http://support.microsoft.com/kb/328912, package om de adodb driver te installeren
[9] http://msdn.microsoft.com/en-us/library/aa289503(VS.71).aspx , .Net heeft de mogelijkheid om files te includen in het project
[10] Firepackage documentatie, AVT Firepackage versie 2.9, te downloaden op http://www.alliedvisiontech.com
[11]http://www.mathworks.com/access/helpdesk/help/toolbox/compiler/index.html?/access/helpdesk/help/ toolbox/compiler/f10-998688.html , Werking van de afhankelijkheids analyse
[12] Handleiding Elo
[13] Sign Language Recognition by combining Statistical DTW and Independent Classification, Jeroen F. Lichtenauer et al. Mei 2008