Automatisches Navigieren in virtuellen Welten

In dieser Semesteraufgabe soll ein virtueller Roboter (vertreten durch ein Stereo-Kamera-Paar) automatisch durch unterschiedliche vorgegebene bzw. selbst zu erstellende Szenarien gesteuert werden.

Die Aufgaben sollen mit Hilfe des Frameworks **WISP** bearbeitet werden, das neben einer frei konfigurierbaren GUI je ein Modul für

- die 2D-/3D-Bildverarbeitung,
- die Steuerung des fahrbaren Roboters i90,
- die Steuerung eines realen oder
- die Steuerung eines virtuellen AIBO sowie für
- AR- und
- Chaos-und-Fraktale-Anwendungen

zur Verfügung stellt.

Wenn Sie die Praktikumsaufgabe daheim vorbereiten wollen, so können Sie sich das komplette Framework über die Homepage des Graphik-Labors herunterladen:

`FBI-Homepage > Labore > Graphische DV und Multimedia > Downloads`

Bevor Sie Ihren ersten Praktikums-Termin wahrnehmen, sollten Sie sich unbedingt mit **WISP** vertraut machen:

- in **..\WISP\Help**: WISP-Hilfe (wisp.chm inkl. FAQs), WISP_mit_MSVC.doc und HowTo's;
- in **..\WISP\Exercises**: Beispiel-Programme.

Im Unterordner **..\WISP\Exercises\VRAIBO\Frame** finden Sie das Rahmenprogramm **frame.cpp**, das Sie so ergänzen sollen, dass die unten beschriebenen Aufgabenstellungen gelöst werden. Dazu müssen Sie die „leere" Methode **evaluate** so ausfüllen, dass der virtuelle AIBO mit Hilfe eines „Dreh-Parameters" (siehe unten), den er durch Auswertung des Eingabebildes ermittelt, automatisch gesteuert wird..

Im Unterordner **..\WISP\Exercises\BV** finden Sie mehrere Unterordner mit Bildverarbeitungsbeispielen, die Ihnen z.B. den Zugriff auf einzelne Pixel (**PixelZugriff.cpp**) oder komplette Segmentierungs-Ansätze (**Segmentierung.cpp** oder **HistereseSchwellwert.cpp**) demonstrieren.

1. Teilaufgabe (vorgegebenes Szenario zur Verfolgung einer Linie)

Das Rahmenprogramm zeigt das Stereo-Bildpaar, das die beiden Kamera-Augen eines Roboters von einer einfachen virtuellen Welt mit einem Rund-Parcours (**ParcourIllustrator**) und mehreren Hindernissen aufnehmen würden. Zusätzlich werden in weiteren Fenstern z.B. die virtuelle Welt samt Standort des Roboters (im Zentrum des Fensters) aus der Vogelperspektive (Blick auf die x-z-Ebene) und die Szene in Third-Person-Ansicht dargestellt.

Nach dem Programmstart bewegt sich der Roboter im automatischen Modus parallel zu einer der geraden Seiten des Parcours. Alternativ können Sie den Roboter auch mit Hilfe des Navigation-Panels steuern. Ihre Aufgabe besteht nun darin, den Roboter mittels Bildverarbeitung automatisch so zu steuern, dass er dem Parcours möglichst exakt folgt, ohne dabei mit Hindernissen zu kollidieren.

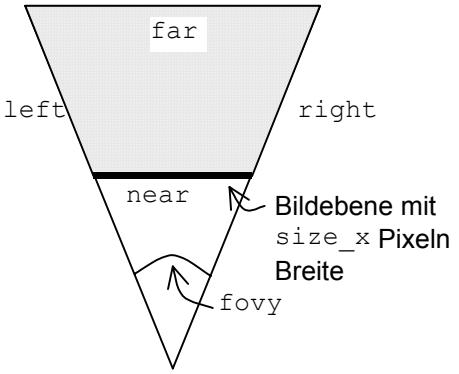
TIPPS:

1.) Der Roboter "blickt" mit zwei Stereo-Kameras, die waagrecht und mit parallel verlaufenden Sehachsen ausgerichtet sind. Der halbe Augenabstand, die Augenhöhe und vieles andere kann gesetzt werden. Alle Größenangaben, wie z.B. die aktuelle Schrittweite des Roboters, aber auch die Abmessungen des Parcours und der Hindernisse, sind in Metern.

2.) Setzen Sie zunächst den Wert d für den halben "Augenabstand" der beiden Stereo-Kameras auf 0 (Methode: **showViews**) und betrachten Sie im Weiteren nur noch das Bild der linken Kamera, die jetzt "mittig" sitzt. Werten Sie den Farbwertverlauf entlang einer waagrecht im Bild verlaufenden Teststrecke aus und ermitteln Sie die Bildposition, an der die Teststrecke die Mitte des Parcours schneidet. Führen Sie einen "Schritt" in diese Richtung durch, werten Sie das von der neuen Roboterposition aus aufgenommene Bild der virtuellen Kamera aus, usw..

3.) Unter Umständen sollten Sie statt des RGB- das HSV-Farbmodell verwenden: **BV::Filter2D::ColorSpaceTransformationFilters**.

4.) Das Frustum ist u.a. durch den waagrechten Öffnungswinkel `fovy` definiert; (siehe Skizze mit Frustum aus der Vogelperspektive). Wenn das Bild `size_x` Spalten hat, so entspricht die waagrechte Ausdehnung eines Pixels an allen Bildpositionen näherungsweise $\alpha = \texttt{fovy} / \texttt{size\_x}$ Grad. Wenn die in einem Bild ermittelte neue Zielposition (z.B. Mittelpunkt des Schnittes der Teststrecke mit dem Parcours) also um `diff_x` Bildpunkte links oder rechts von der Bildmitte liegt, so muss die Kamera vor dem nächsten Vorwärtsschritt um den Winkel $\varphi = \pm\alpha * \texttt{diff\_x}$ gedreht werden.



Nach der Drehung kann ein "Schritt" des Roboters in die neue Richtung durchgeführt werden.

2. Teilaufgabe (Wettbewerb mit selbst definierten Parcours-Teilen)

Jede Praktikums-Zweiergruppe bekommt einen Quadranten des Parcours zur eigenen Gestaltung zugewiesen. Jede Gruppe soll am Ende mit Ihrem virtuellen Roboter den gesamten Parcours automatisch ablaufen können, d.h. die selbst konstruierten, aber auch die von den anderen Gruppen in deren Quadranten eingebrachten Hindernisse bewältigen können. Dabei sind folgende Spielregeln zu beachten:

1.) Die Kontaktstellen des Parcours zu den anderen Quadranten müssen beibehalten werden und der Parcours darf den jeweiligen Quadranten nicht verlassen.

2.) Alle Hindernisse müssen ohne Zusatzwissen zu bewältigen sein; insbesondere darf kein "Insiderwissen" (z.B. die nur den Konstrukteuren bekannte Ausdehnung eines Hindernisses) verwendet werden.

3.) Alle Parcours-Teile müssen den anderen beteiligten Gruppen nach allen Änderungen unverzüglich zugänglich gemacht werden. Ab einem (noch festzulegenden) Stichtag wird das Gesamt-Szenario "eingefroren", d.h. es dürfen keine weiteren Änderungen an den Parcours-Teilen mehr vorgenommen werden.

Ausarbeitung / Abgabe

- Spätestens eine Woche nach Ihrem 2. und Ihrem 4. Praktikums-Termin sollen Sie in gedruckter Form einen kurzen Zwischenbericht abgeben, in dem Sie Ihren aktuellen Stand und Ihre weiteren Absichten einschließlich Zeitplan schildern.
- Spätestens eine Woche nach Ihrem letzten Praktikums-Termin sollen Sie (in einer Klarsichthülle) einen Abschlussbericht abgeben, in dem Sie den erreichten Stand und das dabei eingeschlagene Vorgehen (und insbesondere Ihre Highlights) ausführlich schildern. Legen Sie bitte auch eine CD bei, die alle Ihre gut kommentierten Programm-Listings, die Solution- und Projekt-Dateien und ein ablauffähiges Programm enthält.