King Fahd University of petroleum and minerals
Collage of Computer Science and Engineering

COE 484 - Robotics

Report on Motion Control modules

GT 2005 is a great Simulator for AIBO RoboCup league. It is important to investigate whether parts of the code could be used to program a Kondo humanoid robot. Our goal is to program a Kondo smart enough that it could play soccer against other robots.

Zuhair Y. Khayyat
4/7/2008

1. Objective:

The objective of this report is to identify the relationship between MotionControl module and JointDataBuffer while analyzing how MotionControl interacts with MotionRequest and HeadMotionRequest. There are several classes that represents HeadControl module which are:

DebugMotionControl.cpp

DebugMotionControl.h

GT2005MotionControl.cpp

GT2005MotionControl.h

MotionControl.h

MotionControlSelector.h

MotionStabilizer.cpp

MotionStabilizer.h

WakeUpEngine.cpp

WakeUpEngine.h

2. MotionControl.h:

This header is considered a generic class for motion control modules and it is responsible for setting values for the joints of the robot. The following data are needed by the constructor:

1. Reference to the frame number.
2. Request from the behavior control.
3. Head joint values from the head control.
4. The current body sensor data.
5. The current body posture.
6. walking parameter sets to be used by the WalkingEngine

7. Indication that the Motion Process received a new SensorDataBuffer.
8. Buffer of joint data.
9. The current PID servo gains.
10. Odometry value
11. The height of the neck and the body tilt to be calculated
12. Specifies if the head is blocked by a special action or walk
13. GT2005WalkingEngine

The following classes are needed by MotionControl.h:

- Tools/Module/Module.h
- Representations/Motion/HeadMotionRequest.h
- Representations/Motion/JointDataBuffer.h
- Representations/Motion/PIDData.h
- Representations/Motion/OdometryData.h
- Representations/Motion/MotionInfo.h
- Representations/Perception/SensorDataBuffer.h
- Representations/Perception/BodyPosture.h
- Modules/WalkingEngine/GT2004ParameterSet.h
- Modules/WalkingEngine/GT2005Parameters.h
- Modules/WalkingEngine/GT2005DebugData.h

a) Jointdatabuffer.h:

This header file controls the Number of frames in the buffer where this number defines how many frames are computed each run. In other words, Jointdatabuffer.h controlles how often "Motion::execute" class will be called. JointDataBuffer.cpp is simply

an Implementation of class JointDataBuffer.h. JointDataBuffer.h has two streaming operators to read an write JointDataBuffer data from an to a stream. JointDataBuffer.h imports “Tools/Streams/InOut.h" and "JointData.h".

b) Joindata.h :

It is considered as a data structure that contains the values for all used joints and their names. It has Streaming operator that reads from a stream and writes to a stream JointData data. It implements “Tools/Streams/InOut.h” and “Tools/Streams/Streamable.h”. Joindata.cpp is simply an implements Jointdata.h where Jointdata.h can stream data to the following components:

neckTilt	headPan	headTilt
mouth	earL	earR
legFR1	legFR2	legFR3
legFL1	legFL2	legFL3
legHR1	legHR2	legHR3
legHL1	legHL2	legHL3
tailPan	tailTilt	

c) InOut.h:

InOut.h is the Definition of the abstract base classes In and Out used for streaming. It has two main classes which are In and Out. The class Out is the abstract base class for all classes that implement writing into streams while the class In is the abstract base class for all classes that implement reading into streams. Both

classes provides function that handle reading and writing number of bytes while supporting the following data types:

char	unsigned char	short int	unsigned short int
int	long int	unsigned long int	float
string	endl-symbol		

d) Streamable.h:

It is the base class for all types streamed by the StreamHandler.h that streams directly. Streamable.h also includes “Tools/Streams/InOut.h".

e) SensorDataBuffer.h:

This geader acts as a buffer for sensor data sets which contains all frames receiced from sensors at the same time. It can control the maximum number of frames in the buffer. It has Streaming operators that reads and writes a SensorDataBuffer to and from a stream. It includes the following classes:

" SensorData.h"

"Tools/Streams/InOut.h

"Tools/Streams/Streamable.h".

f) SensorData.h:

This header is a class representing a sensor data vector.

It has Streaming operators that reads and writes a SensorDataBuffer to and from a stream. SensorData.h store both the data from each sensor and its refrence value. It includes

"Tools/Streams/InOut.h" and "Tools/Streams/Streamable.h". The sensors that SensorData.h communicate with is as fillowing:

neckTilt	headPan	headTilt	backR
backF	headPsdNear	mouth	chin
egFL1	legFL2	legFL3	pawFL
legHL1	legHL2	legHL3	pawHL
legFR1	legFR2	legFR3	pawFR
legHR1	legHR2	legHR3	pawHR
tailPan	tailTilt	wlan	backM
accelerationX	accelerationY	accelerationZ	head
headPsdFar	bodyPsd		

g) BodyPosture.h:

class represents the robots body percept the robot stands or is crashed. It implements "Tools/Streams/InOut.h". The following values are calculated and stored in this class:

1) neckHeightCalculatedFromLegSensors
2) bodyRollCalculatedFromLegSensors
3) bodyTiltCalculatedFromLegSensors
4) neckHeightProvidedByMotionControl
5) bodyRollProvidedByMotionControl
6) bodyTiltProvidedByMotionControl
7) bodyRollCalculatedFromAccelerationSensors
8) bodyTiltCalculatedFromAccelerationSensors

h) GT2004ParameterSet.h:

It seems that this class controlls the hardware directly that was used in RoboCup 2004. It has values for walking height, walking width which is the distance from the center of the robot, and x-axis center position. It also havs values of height of lifted foot and telt angles for foot movment. It has Streaming operator that reads GT2004ParametersSets and writes the to a stram. GT2004ParameterSet.cpp is sinply an implementation for the header class GT2004ParameterSet.h. GT2004ParameterSet.h implements the following classes:

"InvKinWalkingParameters.h"

"Tools/Actorics/Kinematics.h"

"Tools/Evolution/Individual.h"

"Tools/Math/Pose2D.h"

i) GT2005Parameters.h:

This class is a parameter class for the GT2005 Walking Engine. It is different than 2004 parameters in several walking aspects. it implements several types of walking such as omnidirectional walking, fast turning, smoothly walking forward, walking diagonal forward, smoothly walking backwards, walking fast backwards, fast sidesteps and dash walk. GT2005Parameters.cpp is an implementation for GT2005Parameters.h. it impelents the following classes:

"Tools/Math/Vector3.h"

"Tools/Math/Pose2D.h"

"Representations/Motion/MotionRequest.h"

"GT2005Polygon.h"

"Tools/Debugging/Debugging.h"

j) GT2005DebugData:

This class detects both the average and diffarntal Acceleration for x,y and z axis. GT2005DebugData.cpp is an implemntation for GT2005DebugData.h

3. T2005MotionControl.h:

This class is the default solution for the module MotionControl. It Integrates head joint values from the head control and joint values from walking Engine. It determines the state of MotionControl while detecting last executed Motion and the latest WalkRequest. GT2005MotionControl.cpp is the implementation of the header class. T2005MotionControl.h includes the following classes:

1) "MotionControl.h"
2) "WakeUpEngine.h"
3) "MotionStabilizer.h"
4) "Modules/WalkingEngine/WalkingEngine.h"
5) "Modules/SpecialActions/SpecialActions.h"
6) "Modules/GetupEngine/GetupEngineSelector.h"
7) "Modules/WalkingEngine/InvKinWalkingEngine.h"
8) "Tools/Module/ModuleHandler.h"

a. WalkingEngine.h:

This header is the interfaces of the module WalkingEngine. Walking parameters can be set to be used by the WalkingEngine by any behavior module such as evolution behavior. This class executs

walking type motions and Calculates the next joint data set. It includes the following classes:

1) "Tools/Module/Module.h"
2) "Tools/RobotConfiguration.h"
3) "Representations/Motion/JointData.h"
4) "Representations/Motion/PIDData.h"
5) "Representations/Perception/SensorDataBuffer.h"
6) "Representations/Motion/OdometryData.h"
7) "Representations/Motion/MotionInfo.h"
8) "InvKinWalkingParameters.h"
9) "GT2004ParameterSet.h"
10) "GT2005Parameters.h"
11) "GT2005DebugData.h"

b. InvKinWalkingEngine.h:

This class is a walking engine based on calculation of rectangular foot movement and inverse kinematics. It calculates new leg speeds according to current motion request and relative foot position for each leg. It also calculates current joint data values and calculate current foot positions.

4. MotionStabilizer.h:

This class is considered as a function that tries to stabilize the robot in such a way that no forces other then gravity act on the center of mass. The main idea is to stabilize walking movement of the robot. This class takes all parameters used to generate a motion and alters the joint datas in

order to stabilize the motion. MotionStabilizer.cpp is the implantation of MotionStabilizer.h. The following classes are implemeted:

"Modules/WalkingEngine/WalkingEngine.h"

"Tools/Math/PIDsmoothedValue.h"

"Representations/Perception/SensorDataBuffer.h"