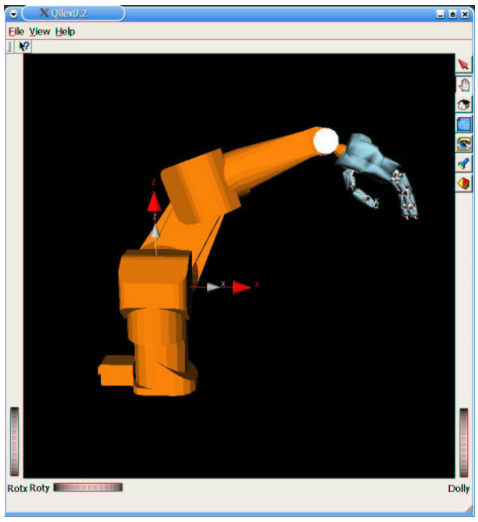

Figura 8.2: Cadena cinemàtica introduïda en la cel·la.

per tractar la informació: init_dat, que té com a entrada un fitxer que conté les dades cinemàtiques de la cadena; i read_element_xml, que s'encarrega de llegir les dades obtingudes a través del mètode init_dat i d'emmagatzemar-les en les diferents estructures pertanyents a la classe Rchain_hand.

- Comprovar que la cel·la no conté cap element amb el mateix nom.
- Inserir al node principal de l'objecte que conté la descripció gràfica una transformació amb la posició i la rotació introduïdes per l'usuari. D'aquesta manera, s'aconsegueix que tota la cadena cinemàtica es vegi afectada per la transformació.
- Es crida la funció insert_kinematic_hand, introduint-li com a paràmetres l'objecte de tipus Rchain_hand inicialitzat amb les dades corresponents i l'objecte que conté la descripció gràfica de la cadena.

Si totes aquestes comprovacions han estat superades amb èxit, la nova cadena cinemàtica és introduïda a la cel·la. En la figura 8.2 es mostra l'aspecte d'una cadena cinemàtica introduïda en l'escena (la mà MA-I acoplada al braç Stäubli RX90).

El mètode doc_insert_kinematic_hand realitza una sèrie d'operacions de certa complexitat que són fonamentals per a la perfecta conjunció dels diferents objectes que conformen l'aplicació final. Les més destacables són les següents:

- Crear els *engines* necessaris per tal de donar moviment a les articulacions i, per tant, aportar animació a l'escena.
- Connectar diferents objectes entre sí per tal que es puguin comunicar. Per exemple, és molt important la connexió de les classes Rchain_hand i SoQtComposeRotation.
- Insertar uns eixos que indiquin el sistema de referència de la base del robot i els sistemes de referència dels extrems dels dits.
- Crear l'objecte que permetrà a l'usuari interactuar amb l'aplicació, és a dir, crear la classe panel_control_hand. A més, es realitzen les diferents connexions entre el panell de control i les altres classes.

Les dues primeres tasques es realitzen primer i de forma simultània, mentre que les altres dues es realitzen després (també simultàniament).

8.4 El resultat final

En la figura 8.3 es pot observar l'aspecte de l'aplicació Qilex0.4 un cop s'han introduït a la cel·la una cadena cinemàtica (el conjunt braç robòtic més mà mecànica) i un objecte geomètric.

Com es pot veure, es distingeixen dos parts clarament diferenciades:

- El panell de control, que permet a l'usuari moure directament les articulacions mitjançant els lliscadors, seleccionar els punts d'aprehensió desitjats per afrontar el problema invers o visualitzar per consola la informació matemàtica addicional.
- L'escena en la qual es mostra la cel·la de treball. En ella es poden veure els diversos models 3D introduïts i els moviments d'aquells que han estat introduïts com a cadena cinemàtica.

Per altra banda, en la figura 8.4 s'observen el conjunt braç més mà i un objecte geomètric després de realitzar el càlcul de la cinemàtica inversa. Com es pot comprovar, en aquest cas s'ha trobat solució i, per tant, el robot ha aconseguit dur a terme amb èxit l'aprehensió de l'objecte.

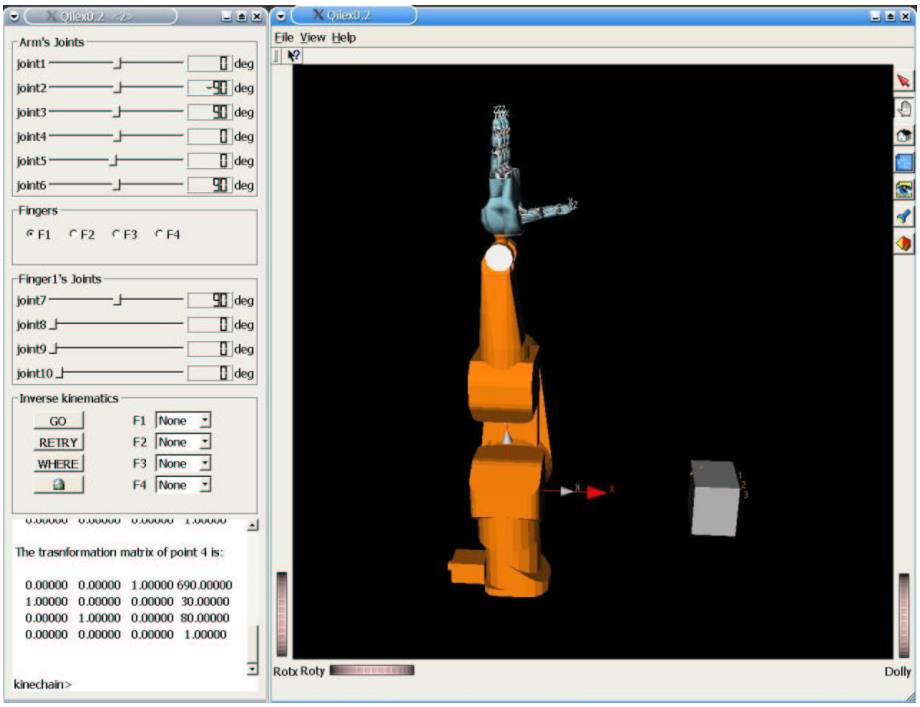

Figura 8.3: L'aplicació final Qilex0.4.

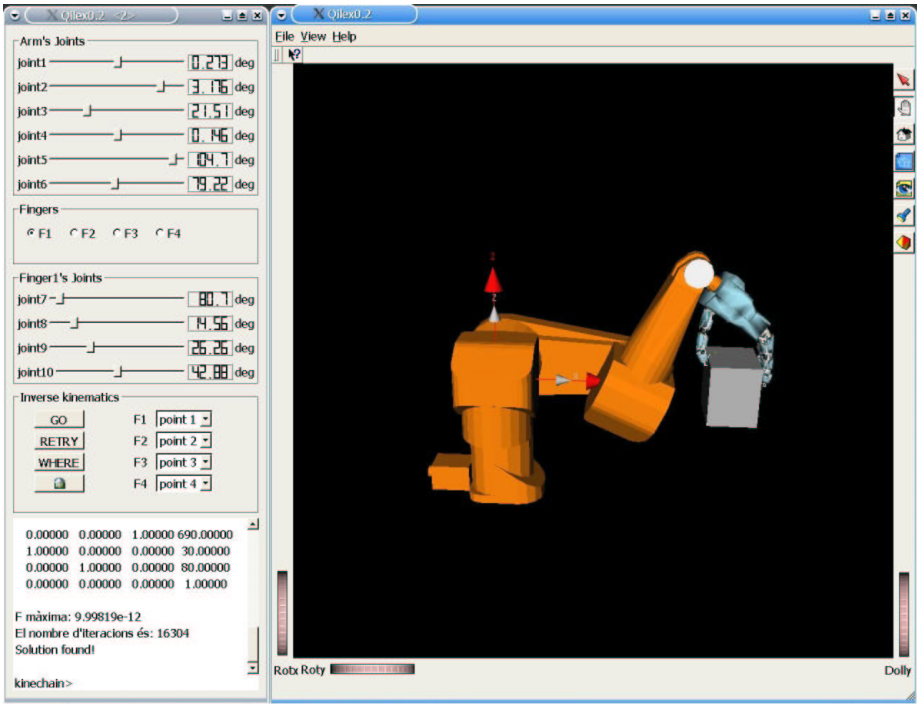

Figura 8.4: L'aplicació final Qilex0.4.

Capítol 9

Pressupost

Per tal de realitzar la valoració econòmica del projecte es tenen en compte per separat les hores destinades a les diferents tasques: investigació, estudi i disseny de l'aplicació, programació i redacció del document.

Concepte	Cost unitari (€/hora)	Unitats (hores)	Subtotal (€)
Cost de personal			
Enginyer	36	400	14.400
Investigador	32	200	6.400
Programador	30	200	6.000
Administratiu	12	60	720
Cost de material			
Amortització equip informàtic	0,04	700	28
		Total	**27.548 €**

El càlcul del cost de l'ús de l'equip es realitza a partir del preu de compra tenint en compte que l'amortització en un centre com l'Institut d'Organització i Control es calcula a 3 anys. Així l'estació de treball té un preu de 1.200 € i l'amortització es de 400 €.

Conclusions

Per tal de construir satisfactòriament l'aplicació final s'han hagut d'afrontar molts problemes i d'assimilar una gran varietat de conceptes, que han fet que la realització d'aquest projecte esdevingués especialment enriquidora. El resultat final obtingut no és només un programa elaborat amb C++, sinó que al darrere hi ha molt més: l'estudi cinemàtic del robot, la búsqueda de solucions davant del problema cinemàtic invers, l'obtenció dels diferents models gràfics, la construcció de l'aplicació amb l'ajuda de Qt, la conjunció de tot plegat... Tot això ha fet que les dimensions i l'abast del projecte hagin acabat essent considerables. Tot i així, el simulador obtingut no preten comparar-se amb els potents simuladors comercials existents ja que aquests acostumen a ser el resultat de la feina d'un equip de gent durant molts anys. El que preten en realitat aquest projecte és donar un pas més per tal de millorar i engrandir el projecte Qilex original.

Per altra banda, val la pena reflexionar sobre com s'ha arribat a assolir un mètode correcte i eficient per tal de resoldre la cinemàtica inversa del conjunt braç robòtic i mà mecànica. El mètode DOM ampliat a quatre cadenes, que és l'encarregat de resoldre aquest problema, no és una simple ampliació del mètode DOM original, sinó que ha estat conseqüència d'hores i hores de proves i de la recerca constant de noves alternatives quan han sorgit problemes. De fet, s'han superat algunes etapes especialment difícils, en les quals resultava complicat trobar respostes i la possibilitat de trobar un mètode vàlid per resoldre la cinemàtica inversa es difuminava en la incertesa. En el fons, doncs, aquest projecte ha estat un treball de recerca que ha permès a l'autor conèixer de primera mà les dificultats i el mèrit del món de la recerca i de la investigació.

El fet que aquest treball no sigui res més que una pedra més afegida en el projecte Qilex ja indica que la idea original és construir un simulador cada cop més gran i amb més i millors funcionalitats. Per tant, és important establir les vies de treball a seguir per continuar millorant el simulador.

Pel que fa al mètode DOM, seria interessant intentar trobar millores que augmentin encara més la seva eficiència. En aquest sentit, un dels punts que podrien ser tractats és el de la funció de distància. En el DOM s'utilitza la norma de Frobenius per tal de determinar la distància entre dues matrius de transformació. Aquesta norma no és del tot correcta, però de moment és útil ja que permet obtenir expressions força simples. Tot i així, és important buscar alternatives a aquesta norma.

Pel que fa a l'estructura de l'aplicació a nivell de classes pot ser útil realitzar una reestructuració general de manera que s'obtinguin diferents mòduls clarament diferenciats. Ara mateix, existeixen múltiples connexions entre diferents classes que dificulten cada cop més el creixement lògic i ordenat de l'aplicació.

Per últim, pel que fa a les funcionalitats de l'aplicació final, cal dir que encara hi ha moltes coses per millorar. Algunes de les idees que es proposen per millorar el producte final són, doncs:

- Dissenyar un intèrpret capaç de desenvolupar diversos mòduls d'ajuda per a la programació, com per exemple generadors de trajectòries amb restriccions geomètriques.
- Desenvolupar nous tipus d'elemets mòbils, a part de les cadenes cinemàtiques.
- Incorporar un sistema de detecció de movimets de manera que quan el robot impactés amb quelcom, s'aturés.
- Potenciar l'ús del *mouse* per tal d'indicar els punts d'aprehensió que es volen assolir ja que es tracta d'una eina ràpida i directa.

Agraïments

A Leo, per la seva paciència i comprensió davant l'allau de dubtes i qüestions que ha rebut per part meva. Sense la seva col·laboració, res del que s'ha fet hagués estat possible.

A Jan, per donar-me la possibilitat de participar en aquest projecte i per la seva ajuda constant en els moments en els quals resultava complicat trobar respostes.

A Greta, Mª Isabel i Cristina, pel seu sacrifici i la seva estima imprescindible en tot moment.

A la resta de família i als meus amics, pel seu recolzament durant tots aquests anys.

I, de forma molt especial, a la persona que amb més il·lusió hagués viscut aquests moments, al meu avi Bernardo, per haver-me ensenyat, entre moltes d'altres coses, que amb esforç i humilitat qualsevol repte és assolible.

Bibliografia

Referències bibliogràfiques

[1] L. Palomo and J. Rosell, "Programari lliure per a la simulació de robots de *n* graus de llibertat," projecte final de carrera, ETSEIB, Barcelona, 2003.

[2] J. Rosell, X. Sierra, L. Palomo, and R. Suárez, "Inverse kinematics for an industrial robot equipped with a dexterous hand." enviat a International Conference on Robotics and Automation, 2004.

[3] X. Sierra, L. Palomo, J. Rosell, and R. Suárez, "Simulador cinemàtic per a l'aprehensió destra d'objectes." enviat a Jornades de Recerca en Automàtica, Visió i Robòtica, 2004.

[4] Institut d'Estudis Catalans, *Diccionari de la Llengua Catalana*. Enciclopèdia Catalana, Edicions 62, 1995.

[5] R. Suárez and P. Grosch, "Mano mecánica MA-I," in *XXIV Jornadas de Automática de León*, 2003.

[6] M. Ardila and R. Gallart, "Diseño e implementación del sistema de control de una mano mecánica," projecte final de carrera, ETSEIB, Barcelona, 2003.

[7] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robótica: control, detección, visión e inteligencia*. McGraw Hill, 1990.

[8] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, pp. 215–221, June 1955.

[9] J. J. Craig, *Introduction to Robotics: mechanical and control*. Addison–Wesley, 1989.

[10] F. B. Ouezdou, S. Regnier, and C. Mavroidis, "Kinematic synthesis of manipulators using a distributed optimization method," *Journal of Mechanical Design, Transactions of the ASME*, vol. 121, no. 4, pp. 492–501, 1999.

[11] S. Regnier, F. Ouezdou, and P. Bidaud, "Distributed method for inverse kinematics of all serial manipulators," *Journal of Mechanism and Machine Theory*, vol. 32, Oct. 1997. `http://citeseer.nj.nec.com/rgnier97distributed.html`.

[12] D. Manocha and Y. Zhu, "A fast algorithm and system for the inverse kinematics of general serial manipulators," in *IEEE International Conference on Robotics and Automation*, pp. 3348–3354, 1994.

[13] F. C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design," *Journal of Mechanical Design, Transactions of the ASME*, vol. 117, pp. 48–54, march 1995.

[14] W. Khalil and J. F. Kleinfinger, "A new geometric notation for open and cloed-loop robots," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, no. pp. 1174-1179, 1986.

[15] A. Law and W. Kelton, *Simulation Modeling and Analysis*. McGRAW-HILL, 1991.

[16] L. J. Aguilar, *C++ a su alcance*. McGraw-Hill, 1991.

[17] J. Wernecke, *The Inventor mentor : programming object-oriented 3D graphics with Open Inventor, release 2*. Addison-Wesley, 1994.

[18] J. A. Mora and S. Domene, "Diseño de una mano de robot para la manipulación diestra de objetos," projecte final de carrera, ETSEIB, Barcelona, 1996.

[19] D. K. Schneider and S. Martin-Michiellot, *VRML Primer and Tutorial*. TECFA, University of Geneva, March 1998.
`http://tecfa.unige.ch/`.

[20] "Pro/DESKTOP software."
`http://www.ptc.com/products/`, 2003. Visitat el març de 2003.

[21] "Trolltech: Qt overview."
`http://www.trolltech.com/products/qt/index.html`, 2003. Visitat l'octubre de 2003.

[22] D. Solin, *Qt Programming in 24 hours*. Sams, 2000.

[23] "Soqt documentation."
`http://doc.coin3d.org/SoQt/`, 2003. Visitat l'octubre de 2003.

[24] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The Int. Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

Bibliografia complementària

- M. Goosesens, F. Mittelbach i A. Samarin. *The LaTeX Companion*. Addison–Wesley, 1994.
- Robert Fuster. *Curs mitjà d'introducció al LaTeX* . Departament de Matemàtica Aplicada, Universitat Politècnica de València, 2000.

Annex A

Paràmetres Denavit-Hartenberg dels robots

A.1 Paràmetres D-H del braç robòtic Stäubli RX90

Seguint la metodologia exposada en el capítol 3, els paràmetres D-H obtinguts per al robot RX90 són els que s'adjunten en la taula següent:

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	-90	0	0	-160	160
2	1	-90	0	450	0	-227.5	47.5
3	1	90	90	0	0	-52.5	232.5
4	1	0	-90	0	450	-270	270
5	1	0	90	0	0	-105	120
6	1	90	0	0	85	-180	360

Taula A.1: Paràmetres D-H del robot RX90.

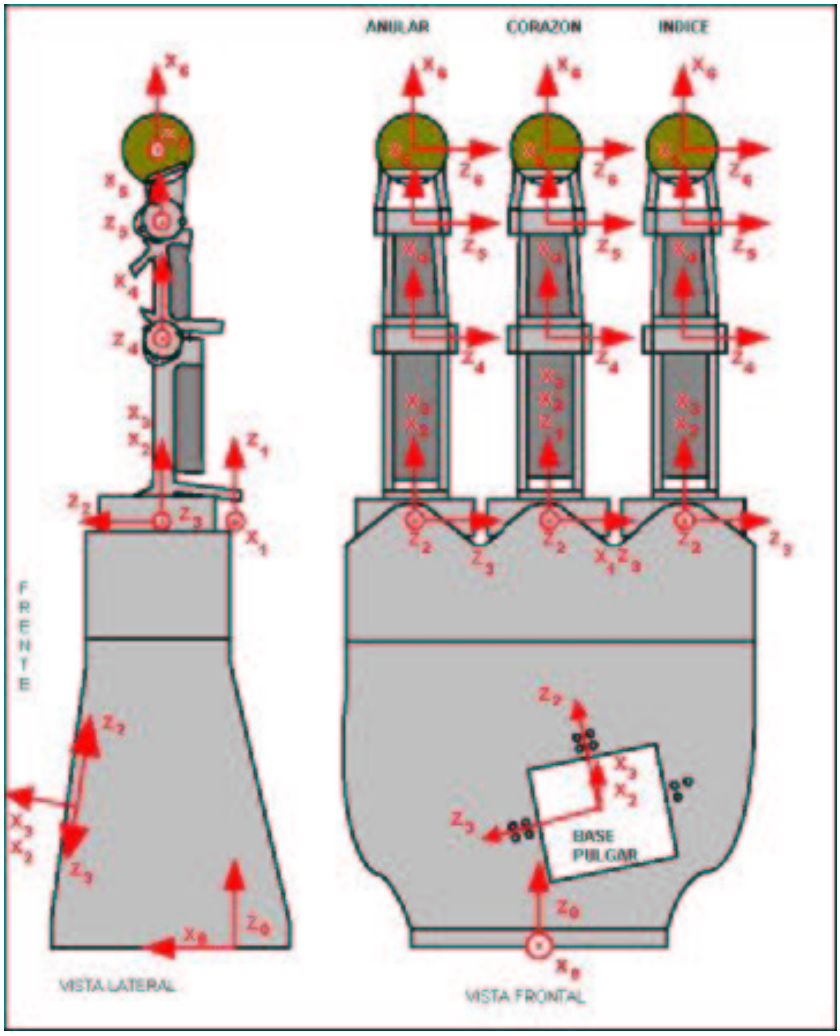

Figura A.1: Sistemes de referència usats per obtenir els paràmetres D-H.

A.2 Paràmetres D-H de la mà mecànica MA-I

Seguint els passos que marca la representació de Denavit-Hartenberg, s'escullen els eixos dels 5 sistemes de referència de cada dit. En el cas de MA-I, es fixa un sistema de referència F_0 a la base de cada dit i solidari a la base de la mà, així com un sistema de referència per cada articulació solidari a cada falange (F_1, F_2, F_3 i F_4). La figura A.1 mostra els diferents sistemes de referència per cada dit. Per una major claretat, en el dibuix no s'han inclòs els eixos de referència del polze, que són iguals que els altres però situats en el pla del dit polze.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	90	0	11	-10	10
2	1	0	0	76	0	0	90
3	1	0	0	56	0	0	90
4	1	0	0	40	0	0	90

Taula A.2: Paràmetres D-H dels dits 0, 1 i 2 (l'anul·lar, el cor i l'índex).

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	90	0	0	-10	10
2	1	0	0	76	0	0	90
3	1	0	0	66	0	0	90
4	1	0	0	45	0	0	90

Taula A.3: Paràmetres D-H del dit 3 (el polze).

Considerant aquests sistemes de referència, els paràmetres D-H de cadascun dels dits s'adjunten en la taules A.2 i A.3. A més, s'hi inclouen els rangs de les diferents articulacions.

A.3 Paràmetres D-H de les articulacions virtuals

Els paràmetres Denavit-Hartenberg de les tres articulacions virtuals que s'afegeixen a cadascun dels dits són els que s'exposen en la taula A.4. Aquests paràmetres són idèntics per als quatre dits.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
5	1	0	-90	0	0	0	90
6	1	-90	-90	0	0	-180	0
7	1	-90	0	0	15	-270	90

Taula A.4: Paràmetres D-H de les articulacions virtuals.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	-90	0	0	-160	160
2	1	-90	0	450	0	-227.5	47.5
3	1	90	90	0	0	-52.5	232.5
4	1	0	-90	0	450	-270	270
5	1	0	90	0	0	-105	120
6	1	90	90	67	276	-180	360
7	1	90	90	0	11	80	100
8	1	0	0	76	0	0	90
9	1	0	0	56	0	0	90
10	1	0	0	40	0	0	90
11	1	0	-90	0	0	0	90
12	1	-90	-90	0	0	-180	0
13	1	-90	0	0	15	-270	90

Taula A.5: Paràmetres D-H del primer dit.

A.4 Paràmetres D-H del conjunt braç més mà

Pot semblar reiteratiu exposar el paràmetres D-H de les quatre cadenes ja que és lícit pensar que aquests paràmetres es poden obtenir directament conjuntant les dades dels apartats anteriors. Això, però, no és del tot cert ja que alguns paràmetres es veuran modificats degut a què al unir braç i mà el sistema de referència inicial canvia. En concret, les articulacions que es veuen modificades són la sisena (l'última del braç) i la setena (la primera de la mà).

Per altra banda, és força interessant presentar aquests paràmetres de forma detallada ja que signifiquen una de les informacions d'entrada de l'aplicació.

Els paràmetres D-H de cadascuna de les quatre cadenes es mostren en les taules A.5, A.6, A.7 i A.8.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	-90	0	0	-160	160
2	1	-90	0	450	0	-227.5	47.5
3	1	90	90	0	0	-52.5	232.5
4	1	0	-90	0	450	-270	270
5	1	0	90	0	0	-105	120
6	1	90	90	0	276	-180	360
7	1	90	90	0	11	80	100
8	1	0	0	76	0	0	90
9	1	0	0	56	0	0	90
10	1	0	0	40	0	0	90
11	1	0	-90	0	0	0	90
12	1	-90	-90	0	0	-180	0
13	1	-90	0	0	15	-270	90

Taula A.6: Paràmetres D-H del segon dit.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	-90	0	0	-160	160
2	1	-90	0	450	0	-227.5	47.5
3	1	90	90	0	0	-52.5	232.5
4	1	0	-90	0	450	-270	270
5	1	0	90	0	0	-105	120
6	1	90	90	-67	276	-180	360
7	1	90	90	0	11	80	100
8	1	0	0	76	0	0	90
9	1	0	0	56	0	0	90
10	1	0	0	40	0	0	90
11	1	0	-90	0	0	0	90
12	1	-90	-90	0	0	-180	0
13	1	-90	0	0	15	-270	90

Taula A.7: Paràmetres D-H del tercer dit.

$Artic.$	$Tipus$	θ_i	α_i	a_i	d_i	$Valor_{min}$	$Valor_{max}$
1	1	0	-90	0	0	-160	160
2	1	-90	0	450	0	-227.5	47.5
3	1	90	90	0	0	-52.5	232.5
4	1	0	-90	0	450	-270	270
5	1	0	90	0	0	-105	120
6	1	56.3	0	72	145	-213.7	326.3
7	1	-56.3	90	0	0	-66.3	-46.3
8	1	0	0	76	0	0	90
9	1	0	0	56	0	0	90
10	1	0	0	40	0	0	90
11	1	0	-90	0	0	0	90
12	1	-90	-90	0	0	-180	0
13	1	-90	0	0	15	-270	90

Taula A.8: Paràmetres D-H del quart dit.

Annex B

Expressions desenvolupades del mètode DOM

B.1 Les matrius que intervenen en el mètode

Tal com s'explica en el capítol 3, el mètode tracta cada articulació de forma separada de manera que s'assigna una distància particularitzada a cada articulació. Aquestes distàncies s'anomenen distàncies locals M_i i s'expressen de la següent manera:

$$M_i = T_{i-1}^{i} \cdot T_i^n - \left(T_0^{i-1}\right)^{-1} \cdot A_0^h$$

Per tal de determinar de forma genèrica l'expressió de M_i, cal anar mostrant el contingut de les diferents matrius que intervenen en el seu càlcul.

La matriu T_{i-1}^{i} és la matriu homogènia de transformació entre dos sistemes de coordenades consecutius i es pot calcular coneixent els paràmetres D-H de la cadena cinemàtica. A més, és l'única que depen de la variable θ_i (o d_i). Val la pena recordar la seva expressió:

$$T_{i-1}^{i} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\theta_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La resta de matrius no depenen de θ_i (o d_i) i es poden expressar com:

$$T_i^n = \begin{bmatrix} t00 & t01 & t02 & t03 \\ t10 & t11 & t12 & t13 \\ t20 & t21 & t22 & t23 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \left(T_0^{i-1}\right)^{-1} \cdot A_0^h = \begin{bmatrix} h00 & h01 & h02 & h03 \\ h10 & h11 & h12 & h13 \\ h20 & h21 & h22 & h23 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Com es pot veure, els termes $t13$, $t23$ i $h23$ són components matricials de T_i^n i $\left(T_0^{i-1}\right)^{-1} \cdot A_0^h$, i, per tant, són valors coneguts.

Per altra banda, si es desenvolupa l'expressió $T_{i-1}^i \cdot T_i^n$ s'obté:

$$T_{i-1}^i \cdot T_i^n = \begin{bmatrix} r00 & r01 & r02 & r03 \\ r10 & r11 & r12 & r13 \\ r20 & r21 & r22 & r23 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

on

$$r00 = \cos\theta_i \cdot t00 - \cos\alpha_i \cdot \sin\theta_i \cdot t10 + \sin\alpha_i \cdot \sin\theta_i \cdot t20$$

$$r01 = \cos\theta_i \cdot t01 - \cos\alpha_i \cdot \sin\theta_i \cdot t11 + \sin\alpha_i \cdot \sin\theta_i \cdot t21$$

$$r02 = \cos\theta_i \cdot t02 - \cos\alpha_i \cdot \sin\theta_i \cdot t12 + \sin\alpha_i \cdot \sin\theta_i \cdot t22$$

$$r03 = \cos\theta_i \cdot t03 - \cos\alpha_i \cdot \sin\theta_i \cdot t13 + \sin\alpha_i \cdot \sin\theta_i \cdot t23 + a_i \cdot \cos\theta_i$$

$$r10 = \sin\theta_i \cdot t00 + \cos\alpha_i \cdot \cos\theta_i \cdot t10 - \sin\alpha_i \cdot \cos\theta_i \cdot t20$$

$$r11 = \sin\theta_i \cdot t01 + \cos\alpha_i \cdot \cos\theta_i \cdot t11 - \sin\alpha_i \cdot \cos\theta_i \cdot t21$$

$$r12 = \sin\theta_i \cdot t02 + \cos\alpha_i \cdot \cos\theta_i \cdot t12 - \sin\alpha_i \cdot \cos\theta_i \cdot t22$$

$$r13 = \sin\theta_i \cdot t03 + \cos\alpha_i \cdot \cos\theta_i \cdot t13 - \sin\alpha_i \cdot \cos\theta_i \cdot t23 + a_i \cdot \sin\theta_i$$

$$r20 = \sin\alpha_i \cdot t10 + \cos\alpha_i \cdot t20$$

$$r21 = \sin\alpha_i \cdot t11 + \cos\alpha_i \cdot t21$$

$$r22 = \sin\alpha_i \cdot t12 + \cos\alpha_i \cdot t22$$

$$r23 = \sin\alpha_i \cdot t13 + \cos\alpha_i \cdot t23 + d_i$$

Finalment, si es resten les expressions $T_{i-1}^{i} \cdot T_{i}^{n}$ i $\left(T_{0}^{i-1}\right)^{-1} A_{0}^{h}$ queda que:

$$M_i = T_{i-1}^{i} \cdot T_{i}^{n} - \left(T_{0}^{i-1}\right)^{-1} A_{0}^{h} = \begin{bmatrix} d00 & d01 & d02 & d03 \\ d10 & d11 & d12 & d13 \\ d20 & d21 & d22 & d23 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

on:

$$d00 = \cos\theta_i \cdot t00 - \cos\alpha_i \cdot \sin\theta_i \cdot t10 + \sin\alpha_i \cdot \sin\theta_i \cdot t20 - h00$$

$$d01 = \cos\theta_i \cdot t01 - \cos\alpha_i \cdot \sin\theta_i \cdot t11 + \sin\alpha_i \cdot \sin\theta_i \cdot t21 - h01$$

$$d02 = \cos\theta_i \cdot t02 - \cos\alpha_i \cdot \sin\theta_i \cdot t12 + \sin\alpha_i \cdot \sin\theta_i \cdot t22 - h02$$

$$d03 = \cos\theta_i \cdot t03 - \cos\alpha_i \cdot \sin\theta_i \cdot t13 + \sin\alpha_i \cdot \sin\theta_i \cdot t23 + a_i \cdot \cos\theta_i - h03$$

$$d10 = \sin\theta_i \cdot t00 + \cos\alpha_i \cdot \cos\theta_i \cdot t10 - \sin\alpha_i \cdot \cos\theta_i \cdot t20 - h10$$

$$d11 = \sin\theta_i \cdot t01 + \cos\alpha_i \cdot \cos\theta_i \cdot t11 - \sin\alpha_i \cdot \cos\theta_i \cdot t21 - h11$$

$$d12 = \sin\theta_i \cdot t02 + \cos\alpha_i \cdot \cos\theta_i \cdot t12 - \sin\alpha_i \cdot \cos\theta_i \cdot t22 - h12$$

$$d13 = \sin\theta_i \cdot t03 + \cos\alpha_i \cdot \cos\theta_i \cdot t13 - \sin\alpha_i \cdot \cos\theta_i \cdot t23 + a_i \cdot \sin\theta_i - h13$$

$$d20 = \sin\alpha_i \cdot t10 + \cos\alpha_i \cdot t20 - h20$$

$$d21 = \sin\alpha_i \cdot t11 + \cos\alpha_i \cdot t21 - h21$$

$$d22 = \sin\alpha_i \cdot t12 + \cos\alpha_i \cdot t22 - h22$$

$$d23 = \sin\alpha_i \cdot t13 + \cos\alpha_i \cdot t23 + d_i - h23$$

B.2 La funció objectiu i els seus termes

Un cop coneguts tots els termes de la matriu M_i, es pot avaluar $\|M_i\|^2$. La seva expressió simplificada és:

$$\begin{aligned}\|M_i\|^2 =& 2\cdot\sin\alpha_i\cdot\cos\theta_i\cdot t20\cdot h10 - 2\cdot\cos\alpha_i\cdot\cos\theta_i\cdot t10\cdot h10 - 2\cdot\sin\alpha_i\cdot\sin\theta_i\cdot t23\cdot h03\\
&+ 2\cdot\cos\alpha_i\cdot\sin\theta_i\cdot t13\cdot h03 - 2\cdot\sin\alpha_i\cdot t10\cdot h20 - 2\cdot\cos\alpha_i\cdot\cos\theta_i\cdot t13\cdot h13\\
&+ 2\cdot\sin\alpha_i\cdot\cos\theta_i\cdot t23\cdot h13 - 2\cdot\cos\alpha_i\cdot\cos\theta_i\cdot t12\cdot h12 + 2\cdot\sin\alpha_i\cdot\cos\theta_i\cdot t22\cdot h12\\
&- 2\cdot\cos\alpha_i\cdot\cos\theta_i\cdot t11\cdot h11 + 2\cdot\sin\alpha_i\cdot\cos\theta_i\cdot t21\cdot h11 - 2\cdot a_i\cdot\sin\theta_i\cdot h13\\
&- 2\cdot\cos\theta_i\cdot t01\cdot h01 - 2\cdot\cos\theta_i\cdot t02\cdot h02 - 2\cdot\cos\theta_i\cdot t03\cdot h03\\
&- 2\cdot a_i\cdot\cos\theta_i\cdot h03 + 2\cdot t03\cdot a_i - 2\cdot\sin\alpha_i\cdot\sin\theta_i\cdot t22\cdot h02\\
&+ 2\cdot\cos\alpha_i\cdot\sin\theta_i\cdot t12\cdot h02 - 2\cdot\sin\alpha_i\cdot\sin\theta_i\cdot t21\cdot h01\\
&+ 2\cdot\cos\alpha_i\cdot\sin\theta_i\cdot t11\cdot h01 + 2\cdot\cos\alpha_i\cdot\sin\theta_i\cdot t10\cdot h00\\
&- 2\cdot\sin\alpha_i\cdot\sin\theta_i\cdot t20\cdot h00 - 2\cdot\sin\theta_i\cdot t03\cdot h13\\
&- 2\cdot\cos\alpha_i\cdot t20\cdot h20 - 2\cdot\sin\alpha_i\cdot t11\cdot h21 - 2\cdot\cos\alpha_i\cdot t21\cdot h21\\
&- 2\cdot\sin\alpha_i\cdot t12\cdot h22 - 2\cdot\cos\alpha_i\cdot t22\cdot h22 + 2\cdot\sin\alpha_i\cdot t13\cdot d_i\\
&- 2\cdot\sin\alpha_i\cdot t13\cdot h23 + 2\cdot\cos\alpha_i\cdot t23\cdot d_i - 2\cdot\cos\alpha_i\cdot t23\cdot h23\\
&- 2\cdot\cos\theta_i\cdot t00\cdot h00 - 2\cdot\sin\theta_i\cdot t00\cdot h10 - 2\cdot\sin\theta_i\cdot t01\cdot h11\\
&- 2\cdot\sin\theta_i\cdot t02\cdot h12 + h00^2 + h03^2 + h02^2 + h01^2 + h10^2 + h20^2 + h13^2 + h12^2\\
&+ h11^2 + h22^2 + h21^2 + t21^2 + t20^2 + t22^2 + t23^2 + t02^2 + t01^2 + t00^2 + a_i^2 + t03^2\\
&+ h23^2 - 2\cdot d_i\cdot h23 + d_i^2 + t12^2 + t13^2 + t10^2 + t11^2\end{aligned}$$

Aquesta expressió és la funció objectiu que es vol minimitzar i és equivalent a:

$$\|M_i\|^2 = dipart + tindep + 2\left(NUM_{\theta_i}\cdot\sin\theta_i + DEN_{\theta_i}\cdot\cos\theta_i\right)$$

ja que:

$$dipart = {d_i}^2 + 2d_i\left(\sin\alpha_i\cdot t13 - h23 + \cos\alpha_i\cdot t23\right)$$

$$
\begin{aligned}
tindep = & -2 \cdot \sin\alpha_i \cdot t10 \cdot h20 + 2 \cdot t03 \cdot a_i - 2 \cdot \cos\alpha_i \cdot t20 \cdot h20 - 2 \cdot \sin\alpha_i \cdot t11 \cdot h21 \\
& - 2 \cdot \cos\alpha_i \cdot t21 \cdot h21 - 2 \cdot \sin\alpha_i \cdot t12 \cdot h22 - 2 \cdot \cos\alpha_i \cdot t22 \cdot h22 \\
& - 2 \cdot \sin\alpha_i \cdot t13 \cdot h23 - 2 \cdot \cos\alpha_i \cdot t23 \cdot h23 + h00^2 + h03^2 + h02^2 + h01^2 \\
& + h10^2 + h20^2 + h13^2 + h12^2 + h11^2 + h22^2 \\
& + h21^2 + t21^2 + t20^2 + t22^2 + t23^2 + t02^2 + t01^2 \\
& + t00^2 + a_i^2 + t03^2 + h23^2 + t12^2 + t13^2 + t10^2 + t11^2 \\
NUM_{\theta_i} = & \cos\alpha_i \cdot t13 \cdot h03 - \sin\alpha_i \cdot t23 \cdot h03 - a_i \cdot h13 - t03 \cdot h13 \\
& - t02 \cdot h12 - t01 \cdot h11 - t00 \cdot h10 - \sin\alpha_i \cdot t20 \cdot h00 \\
& + \cos\alpha_i \cdot t10 \cdot h00 + \cos\alpha_i \cdot t11 \cdot h01 - \sin\alpha_i \cdot t21 \cdot h01 \\
& + \cos\alpha_i \cdot t12 \cdot h02 - \sin\alpha_i \cdot t22 \cdot h02 \\
DEN_{\theta_i} = & -\cos\alpha_i \cdot t13 \cdot h13 - a_i \cdot h03 - t03 \cdot h03 - t02 \cdot h02 - t01 \cdot h01 \\
& - t00 \cdot h00 - \cos\alpha_i \cdot t10 \cdot h10 + \sin\alpha_i \cdot t20 \cdot h10 \\
& + \sin\alpha_i \cdot t21 \cdot h11 - \cos\alpha_i \cdot t11 \cdot h11 + \sin\alpha_i \cdot t22 \cdot h12 \\
& - \cos\alpha_i \cdot t12 \cdot h12 + \sin\alpha_i \cdot t23 \cdot h13
\end{aligned}
$$

Annex C

Els fitxers de dades

L'aplicació final obté totes les dades a partir de fitxers. En concret, hi ha tres tipus de fitxers suportables per part del simulador: els fitxers Inventor/VRML, que contenen la descripció gràfica del robot i dels elements geomètrics; els fitxers amb extensió xml, que descriuen les característiques cinemàtiques dels robots o també poden indicar les coordenades dels punts d'aprehensió introduïts; i els fitxers Qilex de definició d'una cel·la, qlx.

C.1 Els fitxers Inventor/VRML: iv/vrml

Com s'ha dit, els fitxers Inventor/VRML contenen la informació gràfica del robot. Ara mateix, Qilex0.4 pot importar només fitxers gràfics en format Inventor v2.1., però seria interessant i relativament senzill aconseguir que els fitxers amb format VRML v1.0 puguin ser també entenibles per l'aplicació. Cal recordar que Open Inventor és un conjunt d'eines basades en OpenGL que permeten la programació d'objectes en 3D de forma senzilla, mentre que VRML (Virtual Reality Modeling Language) pot ser entès com una extensió visual del format WWW. En el fons, Open Inventor no és res més que VRML1.0.

Pel que fa als objectes geomètrics, aquests no contenen cap particularitat especial i l'estructura del fitxer és la mateixa que la que pugui tenir qualsevol objecte gràfic obtingut amb Open Inventor. Les cadenes cinemàtiques, en canvi, cal que tinguin una estructura i una sèrie de particularitats especials que es descriuen a continuació.

Per poder definir un fitxer en format Inventor de manera que Qilex0.4 el pugui utilitzar com a cadena cinemàtica, cal que segueixi l'estructura general següent:

```
Base {
 ... joint1 ...
        Link1 {
           ... joint2 ...
                   Link2 {
                      ... joint3 ...
                           Link3 {
                              ...
                           }
                   }
            }
      }
```

D'aquesta estructura, cal destacar-ne essencialment tres característiques importants: la informació estrictament geomètrica, la connexió cinemàtica i la pròpia forma jeràrquica de l'estructura.

La informació purament geomètrica està continguda en els nodes que en l'esquema anterior són anomenats Base, Link1, Link2,..., Linkdof. Aquests nodes són nodes de forma i poden estar formats per qualsevol de les primitives bàsiques d'Inventor, com poden ser *Cube*, *Cone*, *Cylinder*, *Sphere*, *VertexShape* ...

Els fitxers dels robots subministrats fan servir *VertexShape*, que correspon a una malla triangular, que és construïda definint primer la geometria (les coordenades dels vèrtex en l'espai) i després la topologia (connectant els vèrtex per tal d'obtenir els diferents triangles).

Les comandes bàsiques d'Inventor usades per tal d'implementar aquest tipus d'informació són les següents:

- *Coordinate3,* que permet definir els diferents punts en l'espai de tres dimensions.
- *IndexedFaceSet,* que s'encarrega de construir els triangles a partir de tres dels punts

definits amb la comanda anterior.

Pel que fa a la connexió cinemàtica, cal dir que es tracta d'una característica vital introduïda en els fitxers per dues raons: per una banda, permet animar o donar moviment a un objecte, que en principi, és estàtic o inanimat i, per l'altra, s'encarrega de realitzar la connexió entre el fitxer Inventor i la resta de l'aplicació.

Aquesta característica està inclosa en nodes de propietats que en l'esquema anterior s'anomenen joint1, joint2,..., jointdof; de manera que tots ells estan definits just abans dels respectius elements (joint1 abans de Link1, joint2 abans de Link2...). Això és així ja que cadascun dels joints representa una de les articulacions del robot i, per tant, cada element ha de girar o desplaçar-se tal com marca la respectiva articulació.

La forma de realitzar aquesta connexió amb la resta de l'aplicació per al cas d'una articulació de rotació és la següent:

```
DEF joint1 Transform {
    Rotation  0 0 1 0.0
    }
```

En aquest cas, el que s'està definint és:

- joint1 és una etiqueta per a identificar la transformació de l'articulació.
- Rotation 0 0 1 0.0, defineix la direcció de l'eix de gir (0 0 1) i l'angle[1] (0.0).

Per al cas d'una articulació de translació, la connexió amb la resta de l'aplicació es realitza de la manera següent:

```
DEF joint2 Transform {
    Translation  100 0 0
    }
```

En aquest cas s'identifica amb l'etiqueta joint2 la transformació que defineix l'articulació de

[1]La biblioteca té implementades les rotacions amb quaternions. L'ús dels *engines* obliga a que si es té una rotació i l'angle inicial és 0, cal posar una volta completa, que són 6.2831 radians.

translació. Defineix l'increment de la posició respecte al punt origen del node (Translation 100 0 0).

D'aquesta manera s'aconsegueix que el programa principal tan sols hagi de cercar les diferents etiquetes que correspondríen a les articulacions i connectar-les a els *engines* corresponents.

Per últim, cal fer esment a la peculiar forma jeràrquica del contingut del fitxer. Com es pot veure a l'esquema inicial, els conjunts joint + link no estan definits de forma consecutiva (un després de l'altre), sinó que cadascun d'ells està dins de l'anterior. Aquest fet és essencial ja que permet que en girar o desplaçar una determinada articulació, la resta d'elements que estan definits a partir d'aquesta es vegin afectats per la variació.

Si els conjunts joint + link estiguessin definits de forma consecutiva i no jeràrquica, el gir o desplaçament d'una articulació afectaria només a l'element d'aquests conjunt, mentre que la resta d'elements no es veurien afectats. Aquest fet s'aprofita per definir en paral·lel els quatre dits després de la sisena articulació del braç.

C.2 Els fitxers de descripció de robots: xml

Els paràmetres cinemàtics del robot estan definits segons l'estructura d'un document XML (*eXtensible Markup Language*). XML és un metallenguatge, és a dir, un llenguatge per definir llenguatges. La seva utilització és deguda, bàsicament, a què ajuda a tenir consistència a l'hora d'intercanviar informació i, a més, perquè hi ha un gran nombre de validadors de documents (la biblioteca Qt en porta un), fet que permet que la seva utilització sigui simple i fàcil. Dels documents XML cal saber que n'hi ha de dos tipus:

- *Ben formats*, aquells que compleixen les especificacions del llenguatge respecte a les regles sintàctiques. L'XML defineix una estructura jeràrquica molt estricta que un document ben format ha de complir.
- *Vàlids*, aquells documents que segueixen la sintaxi de l'XML, i que per tant estan ben formats, i, a més, segueixen una estructura determinada per un DTD (*Document Type Definition*, definició del tipus de document), que estableix la definició gramatical del

document.

Tot seguit es mostra un exemple de la descripció del conjunt format pel braç Stäubli RX90 i la mà mecànica MA-I. Aquest document de descripció d'un robot conté dues parts clarament diferenciades: la primera abarca el text comprès entre *<?xml version...* i la línia que només conté els símbols] > i la segona va des de <kinehand name ... fins al final. La primera part és la definició del document DTD (apartat C.2.1), mentre que la segona part conté pròpiament la informació del robot (apartat C.2.2).

```
<?xml version="1.0" encoding="iso8859−15"?>
<!DOCTYPE kinehand [
<!ELEMENT kinehand (kinechain+)>
<!ATTLIST kinehand name CDATA #REQUIRED
                   chains CDATA #REQUIRED
                   robjoint CDATA #REQUIRED>
<!ELEMENT kinechain (joint+)>
<!ATTLIST kinechain name CDATA #REQUIRED
                    num ID #REQUIRED
                    dof CDATA #REQUIRED>
<!ELEMENT joint EMPTY>
<!ATTLIST joint
            id ID #REQUIRED
            sigma (0 | 1) #IMPLIED
            theta CDATA #REQUIRED
            alpha CDATA #REQUIRED
            ai CDATA #REQUIRED
            di CDATA #REQUIRED
            low_rank CDATA #REQUIRED
            up_rank CDATA #REQUIRED
            max_speed CDATA "30"
            max_acc CDATA "1"
            home CDATA "0">
]>
<kinehand name="RX90&Hand" chains="4" robjoint="6">
<kinechain name="finger0" num="0" dof="13">
```

```
<joint id="f0−1" sigma="1" theta="0" alpha="−90" ai="0" di="0"
low_rank="−160" up_rank="160" max_speed="356" max_acc="5" home="0">
</joint>
<joint id="f0−2" sigma="1" theta="0" alpha="0" ai="450" di="0"
low_rank="−227.5" up_rank="47.5" max_speed="356" max_acc="5" home="−90">
</joint>
<joint id="f0−3" sigma="1" theta="0" alpha="90" ai="0" di="0"
low_rank="−52.5" up_rank="232.5" max_speed="296" max_acc="5" home="90">
</joint>
<joint id="f0−4" sigma="1" theta="0" alpha="−90" ai="0" di="450"
low_rank="−270" up_rank="270" max_speed="409" max_acc="5" home="0">
</joint>
<joint id="f0−5" sigma="1" theta="0" alpha="90" ai="0" di="0"
low_rank="−105" up_rank="120" max_speed="800" max_acc="5" home="0">
</joint>
<joint id="f0−6" sigma="1" theta="0" alpha="90" ai="67" di="276"
low_rank="−180" up_rank="360" max_speed="1125" max_acc="5" home="90">
</joint>
<joint id="f0−7" sigma="1" theta="90" alpha="90" ai="0" di="11"
low_rank="80" up_rank="100" max_speed="200" max_acc="5" home="90">
</joint>
<joint id="f0−8" sigma="1" theta="0" alpha="0" ai="76" di="0"
low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
</joint>
<joint id="f0−9" sigma="1" theta="0" alpha="0" ai="56" di="0"
low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
</joint>
<joint id="f0−10" sigma="1" theta="0" alpha="0" ai="40" di="0"
low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
</joint>
<joint id="f0−11" sigma="1" theta="0" alpha="−90" ai="0" di="0"
low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
</joint>
<joint id="f0−12" sigma="1" theta="−90" alpha="−90" ai="0" di="0"
low_rank="−180" up_rank="0" max_speed="200" max_acc="5" home="−90">
```

```
    </joint>
    <joint id="f0-13" sigma="1" theta="-90" alpha="0" ai="0" di="15"
    low_rank="-270" up_rank="90" max_speed="200" max_acc="5" home="-90">
    </joint>
</kinechain>
<kinechain name="finger1" num="1" dof="13">
    <joint id="f1-1" sigma="1" theta="0" alpha="-90" ai="0" di="0"
    low_rank="-160" up_rank="160" max_speed="356" max_acc="5" home="0">
    </joint>
    <joint id="f1-2" sigma="1" theta="0" alpha="0" ai="450" di="0"
    low_rank="-227.5" up_rank="47.5" max_speed="356" max_acc="5" home="-90">
    </joint>
    <joint id="f1-3" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="-52.5" up_rank="232.5" max_speed="296" max_acc="5" home="90">
    </joint>
    <joint id="f1-4" sigma="1" theta="0" alpha="-90" ai="0" di="450"
    low_rank="-270" up_rank="270" max_speed="409" max_acc="5" home="0">
    </joint>
    <joint id="f1-5" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="-105" up_rank="120" max_speed="800" max_acc="5" home="0">
    </joint>
    <joint id="f1-6" sigma="1" theta="0" alpha="90" ai="0" di="276"
    low_rank="-180" up_rank="360" max_speed="1125" max_acc="5" home="90">
    </joint>
    <joint id="f1-7" sigma="1" theta="90" alpha="90" ai="0" di="11"
    low_rank="80" up_rank="100" max_speed="200" max_acc="5" home="90">
    </joint>
    <joint id="f1-8" sigma="1" theta="0" alpha="0" ai="76" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f1-9" sigma="1" theta="0" alpha="0" ai="56" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f1-10" sigma="1" theta="0" alpha="0" ai="40" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
```

```
    </joint>
    <joint id="f1−11" sigma="1" theta="0" alpha="−90" ai="0" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f1−12" sigma="1" theta="−90" alpha="−90" ai="0" di="0"
    low_rank="−180" up_rank="0" max_speed="200" max_acc="5" home="−90">
    </joint>
    <joint id="f1−13" sigma="1" theta="−90" alpha="0" ai="0" di="15"
    low_rank="−270" up_rank="90" max_speed="200" max_acc="5" home="−90">
    </joint>
</kinechain>
<kinechain name="finger2" num="2" dof ="13">
    <joint id="f2−1" sigma="1" theta="0" alpha="−90" ai="0" di="0"
    low_rank="−160" up_rank="160" max_speed="356" max_acc="5" home="0">
    </joint>
    <joint id="f2−2" sigma="1" theta="0" alpha="0" ai="450" di="0"
    low_rank="−227.5" up_rank="47.5" max_speed="356" max_acc="5" home="−90">
    </joint>
    <joint id="f2−3" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="−52.5" up_rank="232.5" max_speed="296" max_acc="5" home="90">
    </joint>
    <joint id="f2−4" sigma="1" theta="0" alpha="−90" ai="0" di="450"
    low_rank="−270" up_rank="270" max_speed="409" max_acc="5" home="0">
    </joint>
    <joint id="f2−5" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="−105" up_rank="120" max_speed="800" max_acc="5" home="0">
    </joint>
    <joint id="f2−6" sigma="1" theta="0" alpha="90" ai="−67" di="276"
    low_rank="−180" up_rank="360" max_speed="1125" max_acc="5" home="90">
    </joint>
    <joint id="f2−7" sigma="1" theta="90" alpha="90" ai="0" di="11"
    low_rank="80" up_rank="100" max_speed="200" max_acc="5" home="90">
    </joint>
    <joint id="f2−8" sigma="1" theta="0" alpha="0" ai="76" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
```

```
    </joint>
    <joint id="f2-9" sigma="1" theta="0" alpha="0" ai="56" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f2-10" sigma="1" theta="0" alpha="0" ai="40" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f2-11" sigma="1" theta="0" alpha="-90" ai="0" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f2-12" sigma="1" theta="-90" alpha="-90" ai="0" di="0"
    low_rank="-180" up_rank="0" max_speed="200" max_acc="5" home="-90">
    </joint>
    <joint id="f2-13" sigma="1" theta="-90" alpha="0" ai="0" di="15"
    low_rank="-270" up_rank="90" max_speed="200" max_acc="5" home="-90">
    </joint>
</kinechain>
<kinechain name="finger3" num="3" dof="13">
    <joint id="f3-1" sigma="1" theta="0" alpha="-90" ai="0" di="0"
    low_rank="-160" up_rank="160" max_speed="356" max_acc="5" home="0">
    </joint>
    <joint id="f3-2" sigma="1" theta="0" alpha="0" ai="450" di="0"
    low_rank="-227.5" up_rank="47.5" max_speed="356" max_acc="5" home="-90">
    </joint>
    <joint id="f3-3" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="-52.5" up_rank="232.5" max_speed="296" max_acc="5" home="90">
    </joint>
    <joint id="f3-4" sigma="1" theta="0" alpha="-90" ai="0" di="450"
    low_rank="-270" up_rank="270" max_speed="409" max_acc="5" home="0">
    </joint>
    <joint id="f3-5" sigma="1" theta="0" alpha="90" ai="0" di="0"
    low_rank="-105" up_rank="120" max_speed="800" max_acc="5" home="0">
    </joint>
    <joint id="f3-6" sigma="1" theta="0" alpha="0" ai="72" di="145"
    low_rank="-213.7" up_rank="326.3" max_speed="1125" max_acc="5"
```

```
    home="56.3">
    </joint>
    <joint id="f3-7" sigma="1" theta="-56.3" alpha="90" ai="0" di="0"
    low_rank="-66.3" up_rank="-46.3" max_speed="200" max_acc="5"
    home="-56.3">
    </joint>
    <joint id="f3-8" sigma="1" theta="0" alpha="0" ai="76" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f3-9" sigma="1" theta="0" alpha="0" ai="66" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f3-10" sigma="1" theta="0" alpha="0" ai="45" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f3-11" sigma="1" theta="0" alpha="-90" ai="0" di="0"
    low_rank="0" up_rank="90" max_speed="200" max_acc="5" home="0">
    </joint>
    <joint id="f3-12" sigma="1" theta="-90" alpha="-90" ai="0" di="0"
    low_rank="-180" up_rank="0" max_speed="200" max_acc="5" home="-90">
    </joint>
    <joint id="f3-13" sigma="1" theta="-90" alpha="0" ai="0" di="15"
    low_rank="-270" up_rank="90" max_speed="200" max_acc="5" home="-90">
    </joint>
</kinechain>
</kinehand>
```

C.2.1 El DTD

A continuació es detalla el significat de cada línia en la definició del tipus de document de l'exemple mostrat abans.

```
<?xml version="1.0" encoding="iso8859-15"?>
```

És una capçalera on s'especifica que és un document XML, la versió i la codificació. De fet és

opcional, però sempre va bé posar-ho.

```
<!DOCTYPE kinehand [
```

Comença un document del tipus kinehand.

```
<!ELEMENT kinehand (kinechain+)>
```

L'element kinehand conté un o varis elements del tipus kinechain (símbol +).

```
<!ATTLIST kinehand name CDATA #REQUIRED
                   chains  CDATA #REQUIRED
                   robjoint  CDATA #REQUIRED>
```

L'element kinehand conté tres atributs: name, chains i robjoint. Tots tres són dades alfanumèriques i és obligatori posar-los.

```
<!ELEMENT kinechain (joint+)>
```

L'element kinechain conté un o varis elements del tipus joint.

```
<!ATTLIST kinechain num CDATA #REQUIRED
                   name CDATA #REQUIRED
                   dof  CDATA #REQUIRED>
```

L'element kinechain conté tres atributs: num, name i dof. Tots tres són dades alfanumèriques i també és obligatori posar-los.

```
<!ELEMENT joint EMPTY> <!ATTLIST joint
                id  ID #REQUIRED
                sigma  (0 | 1) #IMPLIED
                theta  CDATA #REQUIRED
                alpha  CDATA #REQUIRED
                ai  CDATA #REQUIRED
                di  CDATA #REQUIRED
                low_rank  CDATA #REQUIRED
                up_rank  CDATA #REQUIRED
                max_speed CDATA "30"
                max_acc CDATA "1"
```

```
            home CDATA "0">
]>
```

En aquesta part hi ha la definició de l'element joint. Es tracta d'un element buit, però que conté una sèrie d'atributs. A continuació s'expliquen aquests atributs.

- el camp id, és del tipus ID i requerit. Això implica que els valors no es poden repetir. Serà utilitzat com a identificador de cada articulació.
- el camp sigma només pot pendre els valors 0 o 1: serà 1 si l'articulació és de rotació i 0 si és de translació.
- el camp thetha és el valor de θ dels paràmetres de Denavit-Hartenberg (graus).
- el camp alpha és el valor de α dels paràmetres de Denavit-Hartenberg (graus).
- el camp ai és el valor de a_i dels paràmetres de Denavit-Hartenberg en mm.
- el camp di és el valor de d_i dels paràmetres de Denavit-Hartenberg en mm.
- el camp low_rank és el valor del rang inferior de l'articulació (graus o mm).
- el camp up_rank és el valor del rang superior de l'articulació (graus o mm).
- el camp max_speed és el valor de la velocitat màxima. És un camp opcional, però si no s'especifica té un valor per defecte de 30 (graus/s o mm/s).
- el camp max_acc és el valor de l'acceleració màxima. És un camp opcional, però si no s'especifica té un valor per defecte de 1 (graus/s^2 or mm/s^2).
- el camp home és el valor de l'articulació a la posició de *home*. És un camp opcional, però si no s'especifica té un valor per defecte de 0 (graus o mm).

Un cop definit el DTD, ja es pot introduir la informació sobre el robot.

C.2.2 La informació del robot

A continuació es detallen les diferents inscripcions que hi ha en la segona part del fitxer xml.

```
<kinehand name="RX90&Hand" chains="4" robjoint="6">
```

Significa que s'obre l'element kinehand i que aquest té l'atribut name, amb valor RX90&Hand, l'atribut chains, amb valor 4, i l'atribut robjoint, que val 6. Aquesta informació equival a dir

que s'introdueix una estructura cinemàtica de nom RX90&Hand, formada per quatre cadenes cinemàtiques. Aquestes quatre cadenes cinemàtiques tenen sis articulacions comunes, que són les del braç.

```
<kinechain num="0" name="finger0" dof="13">
```

Significa que s'obre un element kinechain i que aquest té l'atribut num, que val 0, l'atribut name amb valor finger0 i l'atribut dof amb valor 13. Aquesta informació correspon a tenir una cadena cinemàtica de nom `finger0` i de 13 graus de llibertat. Aquesta cadena s'identifica amb el número 0. Cal dir que aquesta línia es repeteix tantes vegades com cadenes cinemàtiques hi ha. Quan s'acaba la definició d'una cadena cinemàtica s'usa </kinechain>.

```
<joint id="1" sigma="1" theta="0" alpha="-90" ai="0" di="0"
  low_rank="-160" up_rank="160" max_speed="1" max_acc="1" home="0">
</joint>
```

Aquí s'està definint l'element joint amb els seus atributs. Ja s'ha parlat del seu significat en l'apartat sobre el DTD.

Aquesta definició es repeteix per cada una de les articulacions de la cadena cinemàtica. Així, si hi ha 13 articulacions, tindren 13 elements joint. Cal remarcar que quan s'acaba un element, s'ha de explicitar la seva finalització, en aquest cas : </joint>.

Amb aquestes línies queden definides les característiques cinemàtiques del robot.

C.3 Els fitxers de descripció de punts d'aprehensió: xml

De la mateixa manera que els fitxers de descripció de robots contenen dues parts (el DTD i la informació del robot), els fitxers de descripció de punts d'aprehensió també tenen aquesta estructura: el DTD i la informació pròpia dels diferents punts d'aprehensió. A continuació, es mostra un exemple d'un fitxer de descripció de punts d'aprehensió. Com és lògic, aquests punts han d'estar especificats a partir d'un objecte geomètric prèviament definit.

```
<?xml version="1.0" encoding="iso8859-15"?>
<!DOCTYPE kineobject [
  <!ELEMENT kineobject (point+)>
```

```
  <!ATTLIST kineobject name CDATA #REQUIRED
                       num_point CDATA #REQUIRED >
  <!ELEMENT point EMPTY>
   <!ATTLIST point
                id ID #REQUIRED
                x CDATA #REQUIRED
                y CDATA #REQUIRED
                z CDATA #REQUIRED
                nx CDATA #REQUIRED
                ny CDATA #REQUIRED
                nz CDATA #REQUIRED>
]>
<kineobject name="cube" num_point="4">
   <point id="1" x="80" y="70" z="40" nx="−1" ny="0" nz="0">
   </point>
   <point id="2" x="80" y="0" z="40" nx="−1" ny="0" nz="0">
   </point>
   <point id="3" x="80" y="−70" z="40" nx="−1" ny="0" nz="0">
   </point>
   <point id="4" x="−80" y="30" z="80" nx="1" ny="0" nz="0">
   </point>
</kineobject>
```

En aquest cas, ja no s'exposa el significat de les línies que conformen el DTD ja que la seva interpretació és anàloga a la de l'apartat anterior. Tot i així, és interessant explicar la interpretació de les inscripcions que apareixen en la segona part del document.

```
<kineobject name="cube" num_point="4">
```

Significa que s'obre l'element kineobject i que aquest té l'atribut name, amb valor cube, i l'atribut num_point, amb valor 4. Aquesta informació equival a dir que s'introdueix un objecte geomètric de nom cube sobre el qual hi ha definits 4 punts d'aprehensió.

```
<point id="1" x="80" y="70" z="40" nx="−1" ny="0" nz="0">
</point>
```

Aquí s'està definint un dels punts d'aprehensió. En concret, el significat dels diferents termes és:

- el camp id, és del tipus ID i requerit. Això implica que els valors no es poden repetir. Serà utilitzat com a identificador de cada punt. Es proposa ordenar-los amb un numeral des de 1 fins a num_point.
- el camp x és la component x de les coordenades del punt.
- el camp y és la component y de les coordenades del punt.
- el camp z és la component z de les coordenades del punt.
- el camp nx és la component x de la normal al punt.
- el camp ny és la component y de la normal al punt.
- el camp nz és la component z de la normal al punt.

C.4 Els fitxers de descripció d'una cel·la: qlx

Els fitxers Qilex, amb l'extensió qlx són també fitxers XML de definició de documents. Per la seva explicació, es passa a mostrar el DTD.

```
<?xml version="1.0" encoding="iso8859-15"?>
<!DOCTYPE qilexfile [<!ELEMENT qilexfile (kineelement+, geomelement*,
  graspelement* )>
<!ELEMENT kineelement (model3d, kinehand)>
<!ATTLIST kineelement name CDATA #REQUIRED
                      kineengine CDATA #REQUIRED
                      pos_x CDATA #REQUIRED
                      pos_y CDATA #REQUIRED
                      pos_z CDATA #REQUIRED
                      pos_rx CDATA #REQUIRED
                      pos_ry CDATA #REQUIRED
                      pos_rz CDATA #REQUIRED
                      pos_angle CDATA #REQUIRED>
<!ELEMENT model3d (#PCDATA)>
<!ATTLIST model3d format CDATA #REQUIRED
                  size CDATA #REQUIRED >
```

```
<!ELEMENT kinehand (kinechain+)>
<!ATTLIST kinehand name CDATA #REQUIRED
                   chains CDATA #REQUIRED
                   robjoint CDATA #REQUIRED>
<!ELEMENT kinechain (joint+)>
<!ATTLIST kinechain num ID #REQUIRED
                    name CDATA #REQUIRED
                    dof CDATA #REQUIRED >
<!ELEMENT joint EMPTY>
<!ATTLIST joint
                id ID #REQUIRED
                sigma (0 | 1) #IMPLIED
                theta CDATA #REQUIRED
                alpha CDATA #REQUIRED
                ai CDATA #REQUIRED
                di CDATA #REQUIRED
                low_rank CDATA #REQUIRED
                up_rank CDATA #REQUIRED
                max_speed CDATA "0.5"
                max_acc CDATA "0.5"
                home CDATA "0">
<!ELEMENT geomelement (model3d)>
<!ATTLIST geomelement name CDATA #REQUIRED
                      pos_x CDATA #REQUIRED
                      pos_y CDATA #REQUIRED
                      pos_z CDATA #REQUIRED
                      pos_rx CDATA #REQUIRED
                      pos_ry CDATA #REQUIRED
                      pos_rz CDATA #REQUIRED
                      pos_angle CDATA #REQUIRED>
<!ELEMENT graspelement (model3d,kineobject)>
<!ATTLIST graspelement name CDATA #REQUIRED
                      pos_x CDATA #REQUIRED
                      pos_y CDATA #REQUIRED
                      pos_z CDATA #REQUIRED
```

```
                    pos_rx CDATA #REQUIRED
                    pos_ry CDATA #REQUIRED
                    pos_rz CDATA #REQUIRED
                    pos_angle CDATA #REQUIRED>
<!ELEMENT kineobject (point+)>
<!ATTLIST kineobject name CDATA #REQUIRED
                     num_point CDATA #REQUIRED >
<!ELEMENT point EMPTY>
<!ATTLIST point
            id ID #REQUIRED
            x CDATA #REQUIRED
            y CDATA #REQUIRED
            z CDATA #REQUIRED
            nx CDATA #REQUIRED
            ny CDATA #REQUIRED
            nz CDATA #REQUIRED>
]>
```

Aquí s'ha definit el DTD del document qilexfile. Aquest conté un o més elements del tipus kineelement i pot contenir cap, un o varis elements dels tipus geomelement i graspelement (simbol *).

C.4.1 kineelement

L'element kineelement representa un robot amb la seva representació gràfica i la seva modelització. Aquest conté dos elements: model3d, kinechain i una sèrie d'atributs. L'element kinechain es defineix igual que l'exposat a l'apartat C.2 que representa les dades per la modelització i l'element model3d representa la descripció gràfica i es descriu a l'apartat C.4.2. Els atributs són:

- name, nom de l'element a l'escena.
- kineengine, identificador del tipus de motor cinemàtic.
- pos_x, pos_y, pos_z, valors de la posició a l'escena.
- pos_rx, pos_ry, pos_rz, pos_angle, valors de l'eix de l'orientació i l'angle. Els valors de

l'angle estan en graus.

C.4.2 model3d

L'element model3d està definit de forma que conté atributs i dades. Els atribus són:

- format, un identificador sobre el tipus d'informació que conté (vrml1, vrml2 ...) i que entén la biblioteca gràfica.
- size, on està la mida en *bytes* del fitxer gràfic.

Les dades que conté són un volcat del fitxer Inventor/VRML amb la descripció gràfica.

C.4.3 geomelement

L'element geomelement representa un element geomètric estàtic. Només conté un element model3d amb la representació i uns atributs. Els atributs son: name, pos_x, pos_y, pos_z, pos_rx, pos_ry, pos_rz i pos_angle i tenen el mateix significat que els definits a l'apartat C.4.1.

C.4.4 graspelement

Per últim, l'element graspelement representa un element geomètric que incorpora punts d'aprehensió. Aquest conté dos elements: model3d, kineobject i una sèrie d'atributs. L'element kineobject es defineix com s'explica en l'apartat C.3 (aporta la informació referent als punts d'aprehensió), mentre que l'element model3d representa la seva descripció gràfica (apartat C.4.2). Els atributs són els mateixos que conté l'element geomelement.

Annex D

El manual de l'usuari

Aquest manual preten ser una guia que expliqui als usuaris de Qilex0.4 com funcionen els diferents menus que conté l'aplicació. Aquest capítol és, per tant, una eina força útil per tal de facilitar i potenciar la plena comprensió del simulador.

D.1 L'inici de Qilex0.4

Per tal d'executar el programa Qilex0.4 cal usar el sistema operatiu GNU/Linux[1]. El primer que apareix després d'executar-lo és una pantalla com la que mostra la figura D.1. Com es pot observar, en l'escena hi apareixen uns eixos de coordenades per defecte. Es tracta del sistema de referència del món o de la cel·la.

La barra del menú té tres ítems:

- *File* serveix per realitzar les operacions relacionades amb fitxers.
- *View* serveix per triar la visualització de la barra d'eines i de la barra d'estat.
- *Help* dóna informació sobre el programa.

En el menú *File* apareixen tres opcions: obrir una cel·la ja definida, construir-ne una de nova o sortir de Qilex0.4.

[1] L'aplicació ha estat implementada usant la distribució Debian de GNU/linux i, de moment, no és possible executar-la en entorns diferents a aquest.

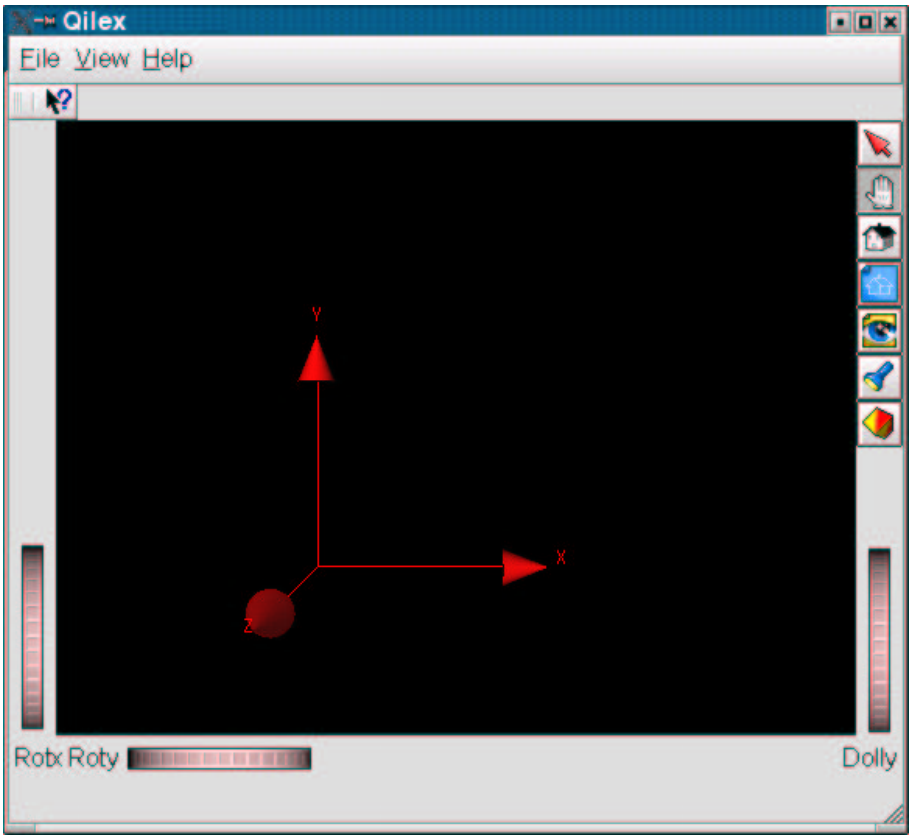

Figura D.1: Pantalla inicial de Qilex0.4.

D.2 La construcció d'una nova cel·la

Si es tria l'opció construir una nova cel·la, la barra de menú permet introduir un nou objecte i es despleguen tres opcions, tal com mostra la figura D.2.

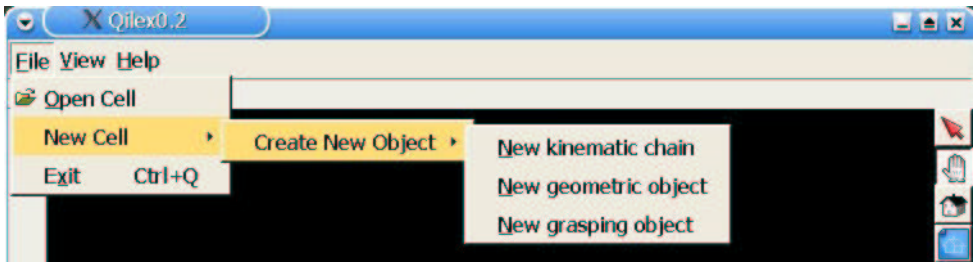

Figura D.2: Barra de menú inicial.

D.2.1 La introducció d'un nou objecte geomètric

Aquesta opció permet a l'usuari introduir peces geomètriques prèviament elaborades. En triar-la, apareix la finestra que mostra la figura D.3.

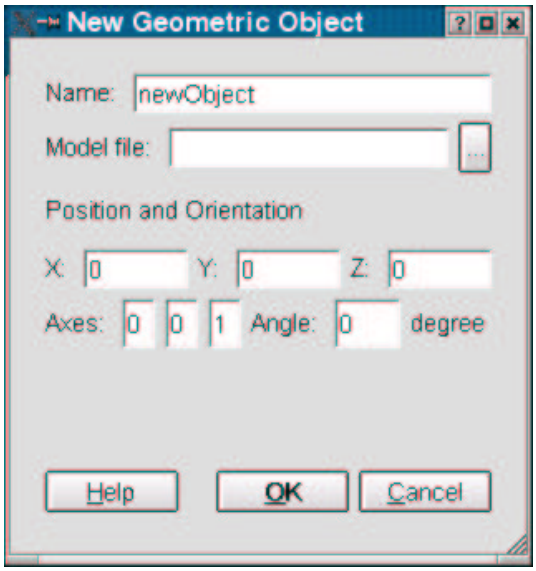

Figura D.3: Finestra per introduir un nou objecte geomètric.

El camp *Name* és obligatori omplir-lo. A més, s'ha de complir que tots els objectes tenen un nom diferent. El camp *Model file* obre un selector de fitxers. Al directori objects de l'arrel de Qilex0.4 hi ha un objecte geomètric definit que porta el nom de *base*. Els valors X, Y i Z representen la posició del centre de coordenades de la peça respecte al centre de coordenades del món. Els valors del camp *Axes* defineixen un vector a l'espai sobre el qual s'aplicarà la rotació d'angle definit al camp *Angle* i que determinarà l'orientació de la peça.

Quan s'ha introduït l'objecte correctament apareix la finestra que mostra la figura D.4.

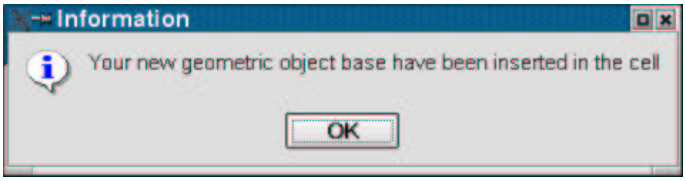

Figura D.4: Finestra informativa.

En la figura D.5 es mostra l'aspecte de la pantalla un cop s'ha introduït l'objecte *base*.

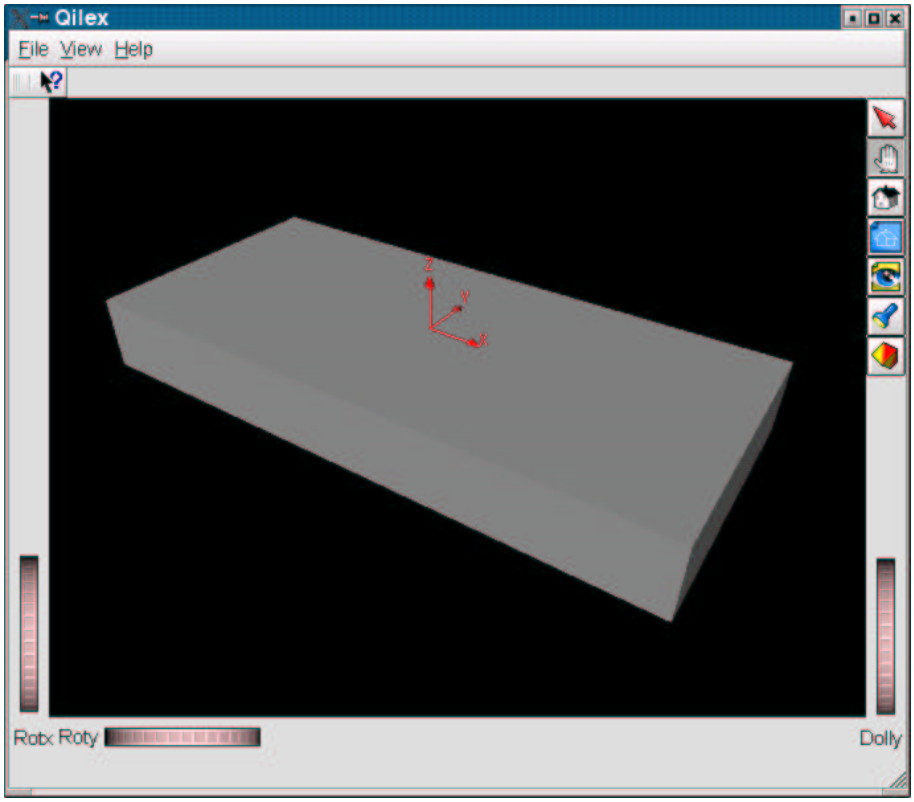

Figura D.5: Pantalla amb l'objecte *base* inserit.

En aquest punt, si es torna a la barra de menú es pot observar que s'han activat noves opcions. Bàsicament, aquestes noves opcions fan referència a la possibilitat de modificar la cel·la(*Edit Cell*), de tancar-la (*Close Cell*) o de guardar-la amb extensió .qlx (*Save Cell*).

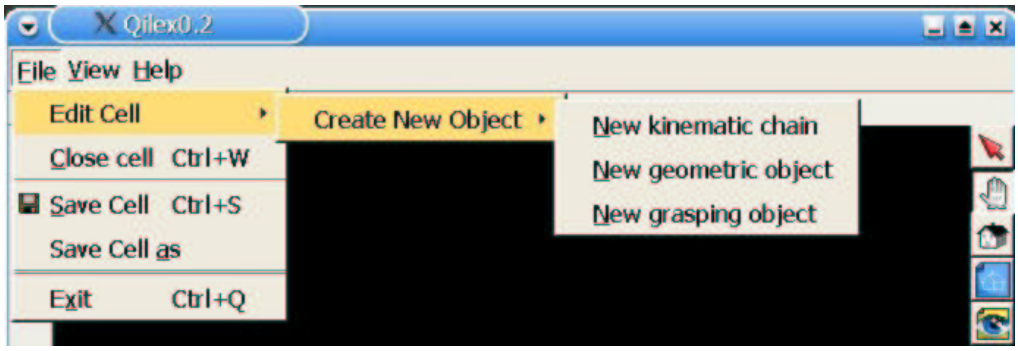

Figura D.6: Barra de menú amb més opcions.

D.2.2 La introducció d'un nou objecte geomètric amb punts d'aprehensió

Aquesta opció permet a l'usuari introduir peces geomètriques que incorporen punts d'aprehensió. En triar-la, apareix la finestra que es mostra en la figura D.7.

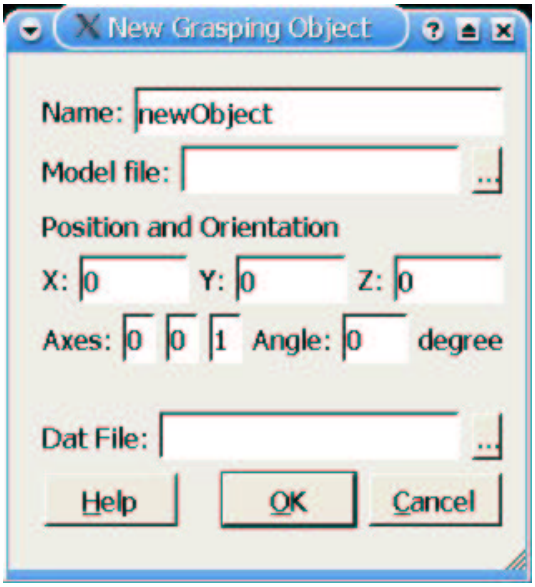

Figura D.7: Finestra per introduir un nou objecte geomètric amb punts d'aprehensió.

Com es pot observar, en aquesta finestra apareixen els mateixos camps que en el cas anterior (introducció d'un objecte geomètric simple) amb l'única excepció de què ara apareix un camp addicional (*Dat file*), que permet seleccionar un fitxer amb la informació referent als punts d'aprehensió. En el directori arrel de Qilex0.4 hi ha un objecte geomètric amb punts d'aprehensió anomenat *cube*.

Quan l'objecte ha estat introduït correctament apareix la finestra mostrada en la figura D.8.

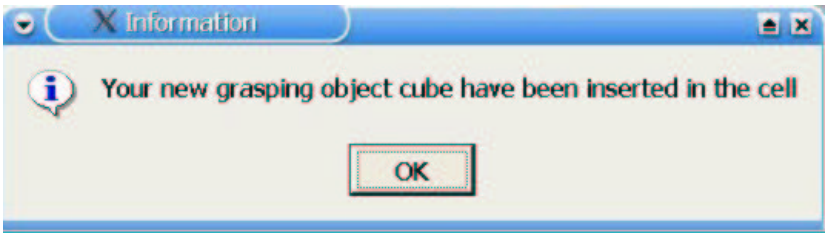

Figura D.8: Finestra informativa.

En la figura D.9 s'observa la pantalla amb l'objecte *cube* inserit.

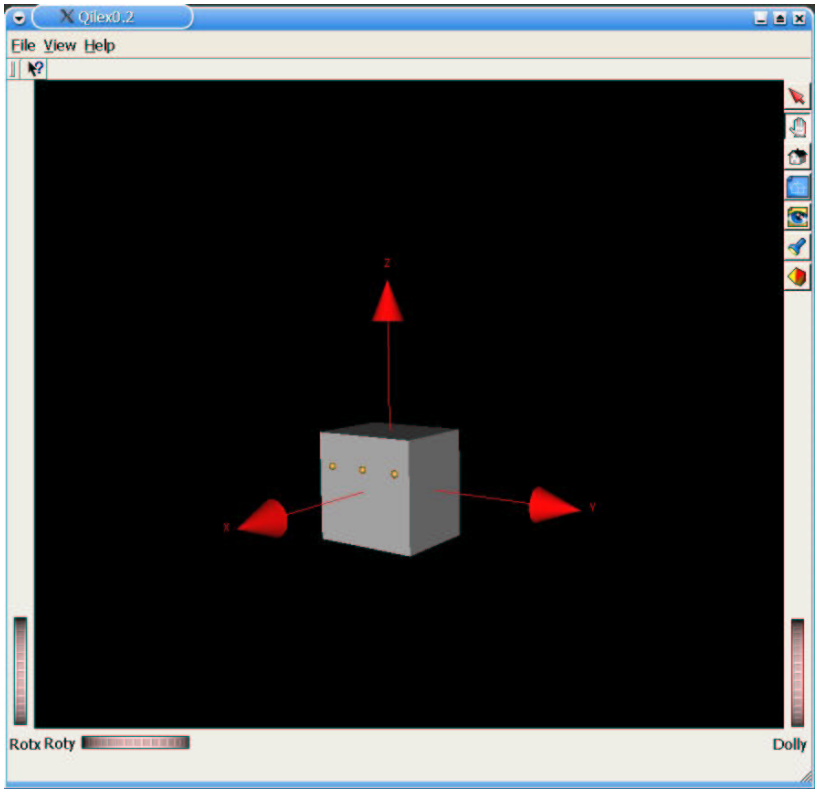

Figura D.9: Pantalla amb l'objecte *cube* inserit.

D.2.3 La introducció d'una nova estructura cinemàtica

Per tal d'introduir en el simulador una estructura cinemàtica com la formada per la mà mecànica MA-I acoplada al robot Stäubli RX90, cal escollir l'opció *New kinematic chain* del submenú *Create New Object* del menú *Edit Cell* (o *New Cell*). La finestra que apareix és la que es mostra en la figura D.10.

En el camp *Name* cal introduir un nom diferent per cada cadena cinemàtica inserida. El fitxer model (*Model file*) ha de ser un fitxer de descripció gràfica de robots. Al directori robots de l'arrel de Qilex0.4 n'hi ha alguns exemples. El fitxer corresponent al conjunt braç més mà és Rx90&Hand.iv. Per altra banda, en els camps de *Position and Orientation* cal introduir la posició i orientació respecte a les coordenades del món de la cel·la. El camp *Kinematic Engine* permet escollir el motor cinemàtic desitjat (Rchain o Rchain_hand). El motor idoni per tractar el conjunt braç més mà és Rchain_hand. Per últim, el camp *Dat file* permet escollir el fitxer

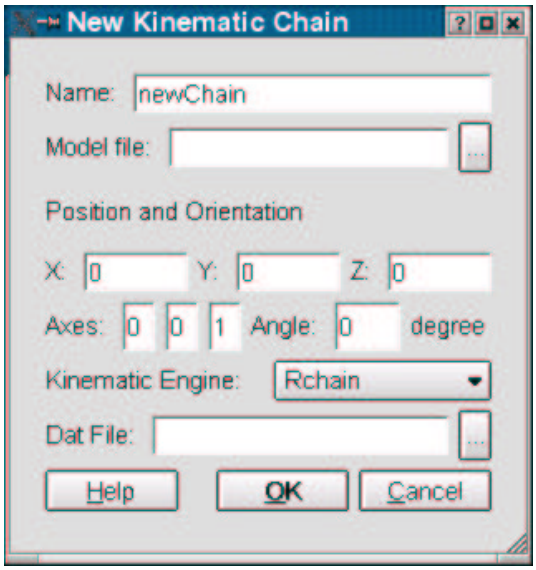

Figura D.10: Finestra per introduir una nova cadena cinemàtica.

que incorpora la descripció cinemàtica del robot. Per al cas del conjunt braç més mà el fitxer a escollir és Rx90&Hand_dat.xml.

Si la cadena cinemàtica ha estat introduïda correctament, apareix la finestra mostrada en la figura D.11.

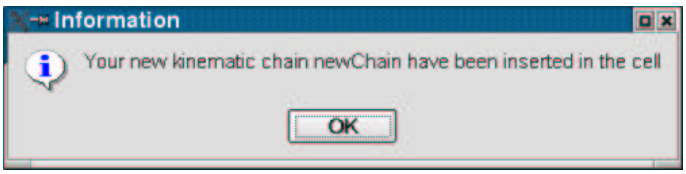

Figura D.11: Finestra informativa.

En la figura D.12 s'observa la pantalla amb l'estructura cinemàtica introduïda. Com es pot veure, a part d'aparèixer el robot en l'escena, també apareix una nova finestra per al comandament del robot: el panell de control.

Es poden insertar tants objectes i tantes cadenes cinemàtiques com calgui. Per cada nova cadena cinemàtica apareixerà un nou panell de control per al seu comandament.

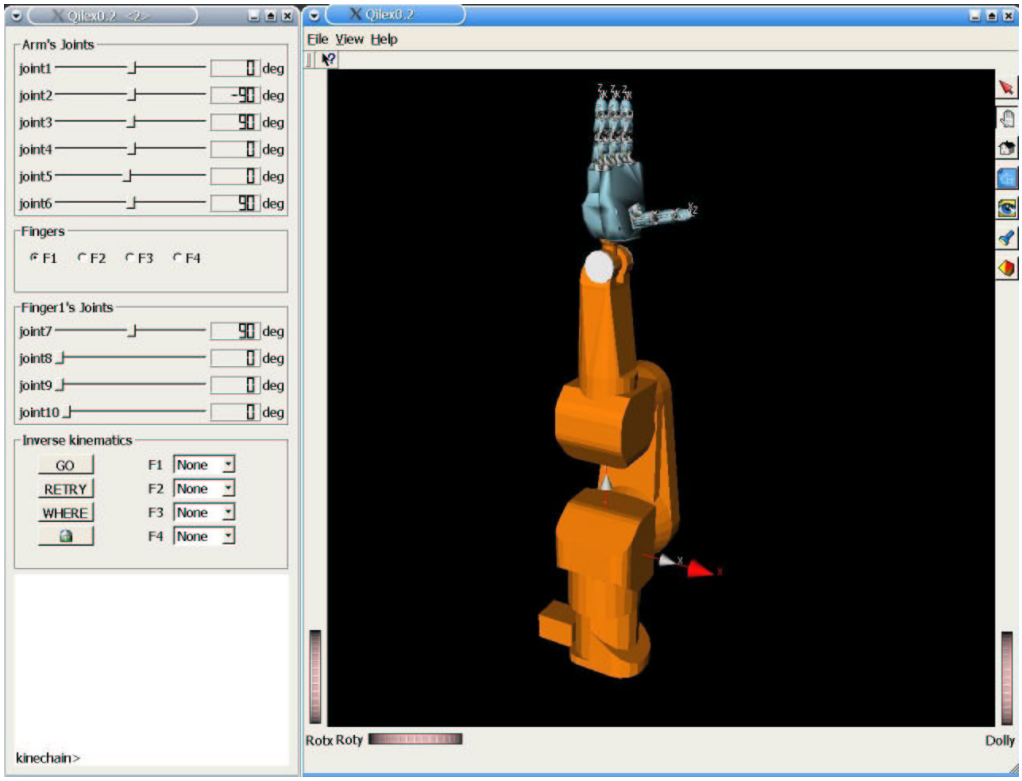

Figura D.12: Pantalla amb el robot RX90 i la mà mecànica insertats.

D.3 La manipulació dels objectes de la cel·la

D.3.1 La manipulació de la càmera

La biblioteca Inventor/Coin té definides unes funcions que permeten moure la càmera amb la que s'observa l'escena. Aquestes funcions modifiquen només la manera de visualitzar l'escena i, per tant, no permeten interactuar amb els objectes geomètrics o les cadenes cinemàtiques. Un exemple d'aquestes funcionalitats són les rodetes *rotx* i *roty* que apareixen a la part inferior esquerra de la pantalla i que produeixen una rotació de l'escena en els eixos de la càmera X i Y, respectivament. Per altra banda, la rodeta *dolly* que apareix a la part inferior dreta de la pantalla realitza un moviment endavant i endarrere de la càmera. Altres funcions d'aquest tipus són les següents:

 Selecciona objectes

 Manipula la càmera

 Porta la càmera a la posició inicial

 Defineix una nova posició inicial de la càmera

 Visualitza tota l'escena

 Focalitza (*zoom*) en el punt que es senyali amb el punter

 Canvia la perspectiva de cònica a isomètrica i viceversa

D.3.2 La manipulació del robot

Per poder interactuar amb el robot cal recórrer al seu panell de control. En concret, el panell de control del braç robòtic més la mà és el que es mostra en la figura D.13.

La implementació del panell de control i les diferents possibilitats que ofereix s'expliquen en l'apartat 6.2.2 de la memòria. Tot i així, val la pena repetir i ampliar algunes de les seves principals característiques per tal d'aconseguir que l'usuari tingui del tot clar com funciona aquesta important eina.

El panell està dividit en tres parts:

- La primera és on hi ha els lliscadors que permeten variar el valor de les articulacions del braç i dels dits.
- La segona part permet realitzar el càlcul de la cinemàtica inversa un cop s'han escollit els punts d'aprehensió que es volen assolir.
- La tercera part és una consola que permet mostrar informació addicional.

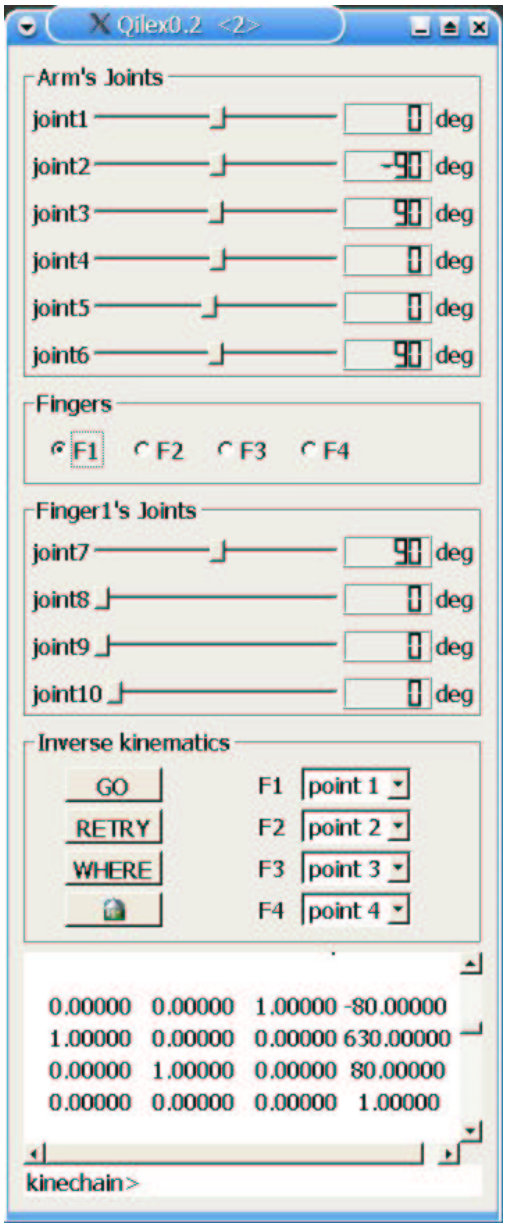

Figura D.13: Panell de control del robot.

En la segona part (*Inverse kinematics*) cal tenir en compte que:

- Els camps *F1*, *F2*, *F3* i *F4* permeten assignar un punt d'aprehensió a assolir per cada dit. Cada punt d'aprehensió té una etiqueta $(1, 2, 3 \ldots)$ que es visualitza en l'escena.
- El botó GO permet iniciar el càlcul de la cinemàtica inversa sempre i quan s'hagin escollit els punts d'aprehensió a assolir.

- El botó RETRY permet tornar a intentar el càlcul de la cinemàtica inversa partint d'una configuració inicial diferent.
- El botó WHERE s'encarrega de mostrar per consola la configuració actual del robot. Per fer-ho, s'imprimeixen en la consola les matrius dels TCP's de cada dit.
- El botó  ubica el robot a la posició de *home* definida en el fitxer xml.

La consola, a part de mostrar les matrius dels TCP's en clicar el botó de WHERE, aporta d'altres informacions força útils. Cal destacar, per exemple, que si s'inicia el càlcul de la cinemàtica inversa i no s'han escollit punts d'aprehensió a assolir, mostra el següent missatge: *No grasping points have been selected*. O, per altra banda, imprimeix *Solution found* o *Solution no found* depenent de si el càlcul de la cinemàtica inversa ha finalitzat satisfactòriament o no.

Si el motor cinemàtic aconsegueix trobar solució, el robot es mou a la configuració obtinguda. La figura D.14 mostra un exemple del resultat que s'obté en aquest cas.

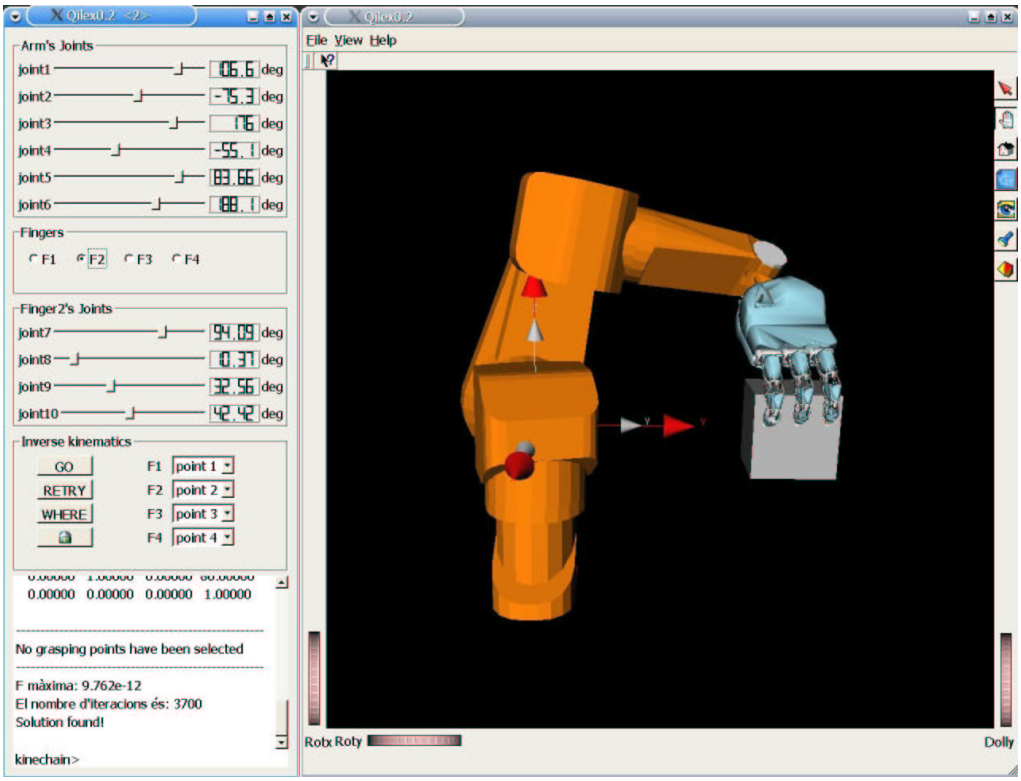

Figura D.14: El programa Qilex0.4 després de resoldre el problema cinemàtic invers.

Annex E

Els fitxers .h

En aquest capítol, s'adjunten els fitxers .h (*header files*) més importants usats en l'aplicació. Cadascun d'aquests incorpora, a més, comentaris encaminats a aclarir el significat de moltes de les variables i funcions definides.

E.1 El fitxer qilex.h

```
// forward declaration of the Qilex classes
class QilexDoc;
class QilexView;

class QilexApp : public QMainWindow
{
  Q_OBJECT
  friend class QilexView;
  public:
    // construtor of QilexApp, calls all init functions to create the
    // application.
    QilexApp(QWidget* parent=0, const char* name=0);
    ~QilexApp();

    // opens a file specified by commandline option
```

```
QilexDoc *getDocument() const;

// initializes all QActions of the application
void initActions ();
// initMenuBar creates the menu_bar and inserts the menuitems
void initMenuBar();
// normalMenuBar creates the normalmenu_bar for normal operation with
// the cell
void normalMenuBar();
void initMenuBar_again();
bool new_kinechain;

// this creates the toolbars. Change the toolbar look and add new
// toolbars in this function
void initToolBar();
// setup the statusbar
void initStatusBar();
// setup the document
void initDoc();
// setup the mainview
void initView_Doc();

// overloaded for Message box on last window exit
bool queryExit();

void completeView(int dof);
void openAcell(const QString &fileName);

public slots:
// open a document
void slotcellOpen();
// save a document
void slotcellSave();
// save a document under a different filename
void slotcellSaveAs();
```

```
  // close the actual file
  void slotcellClose ();
  // print the actual file
  void slotscenePrint ();
  // exits the application
  void slotcellQuit ();
  // put the marked text/object into the clipboard and remove
  // it from the document
  void slotEditCut ();
  // put the marked text/object into the clipboard
  void slotEditCopy ();
  // paste the clipboard into the document
  void slotEditPaste ();
  // toggle the toolbar
  void slotViewToolBar(bool toggle);
  // toggle the statusbar
  void slotViewStatusBar(bool toggle);

  /** shows an about dlg*/
  void slotHelpAbout();

  void slotobjectInsert ();

  void slotnew_kinematic_chain ();
  void slotnew_geometric_object ();
  void slotnew_grasping_object ();

signals :
  void new_grasp_value ( ct_new_grasping_object *);

private :
  // view is the main widget which represents your working area.
  // The View class should handle all events of the view widget.
  // It is kept empty so you can create your view according to your
  // application's needs by changing the view class.
```

```
    QilexView *view;
  // doc represents your actual document and is created only once.
  // It keeps information such as filename and does the serialization
  // of your files.
    QilexDoc *doc;

  // file_menu contains all items of the menubar entry "File"
    QPopupMenu *cellMenu;
  // view_menu contains all items of the menubar entry "View"
    QPopupMenu *viewMenu;
  // view_menu contains all items of the menubar entry "Help"
    QPopupMenu *helpMenu;
    QPopupMenu *cellNew;
    QPopupMenu *cellCreateNewObject;

    QToolBar *fileToolbar;

    QAction *cellOpen;
    QAction *objectInsert;
    QAction *Bnew_kinematic_chain;
    QAction *Bnew_geometric_object;
    QAction *Bnew_grasping_object;
    QAction *cellSave;
    QAction *cellSaveAs;
    QAction *cellClose;
    QAction *scenePrint;
    QAction *cellQuit;
    QAction *viewToolBar;
    QAction *viewStatusBar;
    QAction *helpAboutApp;
    bool p_scene_loaded;

};
```

E.2 El fitxer qilexview.h

```
class QilexView : public QWidget
{
   Q_OBJECT
   friend class QilexDoc;

   public:
    // Constructor for the main view
    QilexView(QWidget *parent = 0, const char *name=0);
    // Destructor for the main view
    ~QilexView();

    // returns a pointer to the document connected to the view instance.
    // Mind that this method requires a QilexApp instance as a parent
    // widget to get to the window document pointer by calling the
    // QilexApp::getDocument() method.
      QilexDoc *getDocument() const;

      SoQtExaminerViewer *p_Viewer;
      void setScene(SoSeparator*);
      void addObjectCell(SoSeparator*);

   protected:
    QilexDoc * cell;

    // contains the implementation for printing functionality
    void print(QPrinter *pPrinter);

};
```

E.3 El fitxer qilexdoc.h

```
class QilexDoc : public QObject
{
```

```
Q_OBJECT
    public:
    // Constructor for the fileclass of the application
    QilexDoc(QWidget *parent, const char *name=0, QilexView *pview=0);
    // Destructor for the fileclass of the application
    ~QilexDoc();
    void newDoc();
    bool save();
    bool saveAs(const QString &filename);
    bool isModified() const;
    QString filename;
    //This function introduces a new sample kinematic chain
    int doc_new_kinematic_chain(ct_new_kinematic_chain *data);
    //This function introduces a new kinematic structure (arm+hand)
    int doc_new_kinematic_hand(ct_new_kinematic_chain *data);
    //This function introduces a new geometric object
    int doc_new_geometric_object(ct_new_geometric_object *data);
    //This function introduces a new grasping object
    int doc_new_grasping_object(ct_new_grasping_object *data);
    int doc_opencell(const QString &filename);

  signals:
    void documentChanged();

protected:
    bool modified;
    SoSeparator *scene;

 private:
    QilexView *view;
    SoSeparator* readFile(const char *filename, int &tipus);
    int writeXML_kineelement(const char *buffer, size_t size, int tipus,
    ct_new_kinematic_chain *data, Rchain *kinechain);
    int writeXML_kineelement(const char *buffer, size_t size, int tipus,
    ct_new_kinematic_chain *data, Rchain_hand *kinechain);
```

```
        int writeXML_geomelement(const char *buffer, size_t size, int tipus,
        ct_new_geometric_object *data);

        bool writeXMLheader();
        int doc_insert_kinematic_chain (QDomElement qilexelement);
        int doc_insert_geometric_object (QDomElement qilexelement);
        int doc_insert_grasping_object (QDomElement qilexelement);
        //These functions are called by doc_new_kinematic_chain and
        //doc_new_kinematic_hand
        int doc_insert_kinematic_chain (Rchain *kineengine,
        SoSeparator *kinechain);
        int doc_insert_kinematic_hand (Rchain_hand *kineengine,
        SoSeparator *kinechain);
};
```

E.4 El fitxer panel_control_hand.h

```
class panel_control_hand : public QWidget
{
    Q_OBJECT

public:
    panel_control_hand ( QWidget* parent = 0, const char* name = 0,
    Rchain_hand *pkinechain = 0 );
    ~panel_control_hand ();

    //There is a Qpanel_joint variable for each slider that appears
    // in the control panel
    Qpanel_joint ** ldial ;

    //The buttons and labels that appear in the control panel
    //are defined as follows
    QLabel* TextLabelf0;
    QLabel* TextLabelf1;
```

```
QLabel* TextLabelf2;
QLabel* TextLabelf3;

int grasp_number;
double **sol;
double *sol1;

QCheckBox* check_world;
QCheckBox* check_tool;
QPushButton* pushButton;
QPushButton* kinematic;
QPushButton* kinematic_where;
QPushButton* kinematic_retry;
QButtonGroup* choice;
QRadioButton* f0;
QRadioButton* f1;
QRadioButton* f2;
QRadioButton* f3;
QComboBox* select_f0;
QComboBox* select_f1;
QComboBox* select_f2;
QComboBox* select_f3;
consoleWidget* TextBrowser;

//A new Rchain_hand variable is defined here
Rchain_hand* kinechain;
//This function update the limits of each joint
void update_limits ();
//The next variables and functions are used in the inverse
//kinematics
Rhmatrix* A_hand;
Rhmatrix* pre;
Rhmatrix* post;
void init_angles ();
void init_angles1 ();
```

```
    void random_angles();

protected:
    QVBoxLayout* mainlayout;

    QGroupBox* dials_arm;
    QGroupBox* dials_f0;
    QGroupBox* dials_f1;
    QGroupBox* dials_f2;
    QGroupBox* dials_f3;
    QButtonGroup* GroupBox;
    QWidget* megagrup;

    QHBoxLayout* inv_f0;
    QHBoxLayout* inv_f1;
    QHBoxLayout* inv_f2;
    QHBoxLayout* inv_f3;

    private:
        QPixmap image0;

    public slots:
        //These slots are connected with the buttons of the
        //control panel
        void slotmove_home();
        void slotupdate_fingers ( ct_new_grasping_object *data );
        void slotgo_inverse ();
        void slotwhere_inverse ();
        void slotretry_inverse ();

    private slots:
        void slot_finger0 ();
        void slot_finger1 ();
        void slot_finger2 ();
        void slot_finger3 ();
```

```
};
```

E.5 El fitxer consolewidget.h

```
class consoleWidget : public QWidget
{
    Q_OBJECT

public:
    consoleWidget( QWidget* parent = 0, const char* name = 0,
     WFlags fl = 0 );
    ~consoleWidget();

   QTextEdit* console;

   QLabel* prompt;

   QLineEdit* input;



protected:
   QVBoxLayout* consoleWidgetLayout;
   QHBoxLayout* inputLayout;

public slots:
   //This function inserts the messages in the console
   void insertLineAtEnd(QString line);

};
```

E.6 El fitxer qpanel_joint.h

```
class Qpanel_joint : public QWidget {
```

```
Q_OBJECT

public:
    //This class constructs the sliders and they are connected
    //with joint
    Qpanel_joint(QWidget *parent=0, const char *name=0);
    ~Qpanel_joint();

    QLabel* label_joint;
    QSlider* slider;
    QLCDNumber* joint_value;
    QLabel* unit;
    //The next functions initialize the slider's value and
    //its rank
    void setLabelJoin(int i);
    void setRankJoin(double, double);
    void setformat(int fmt);
    void setMinValue(double);
    void setMaxValue(double);
protected:
    QHBoxLayout* layout;

//The next slots and signals updates the value of the sliders
//and they are connected with the scene
public slots:
    void setValue(double);
private slots:
    void setValue_int(int);
signals:
    void valueChange(double);

private:
    int i_value;
    double d_value;
    int format; //0 rads ; 1 degree ; 2 mm
```

```
};
```

E.7 El fitxer Rchain_hand

```
class Rchain_hand: public QObject
{
    Q_OBJECT

   private :

      Rhtranslink **pT;
      Rhmatrix **ppremult_T;
      Rhmatrix **ppostmult_T;

      // private temporal vars
      Rhtranslink **ptkT;
      Rhmatrix **ptkpremult_T;
      Rhmatrix **ptkpostmult_T;

      bool widget;
      QTimer *ptimer;

      //final position in a movement
      double **pqf;
      // initial position in a movement
      double **pqi;
      //slope | distance
      double **pslope_path;

      double p_speed;
      double p_acceleration;
      double ptime_f;
      double ptime;
```

```
//These functions update ppremult_T and ppostmult_T variables
void pupdate_T_premult (Rhmatrix ppremult_Tt[], Rhtranslink Tt[],
            int n, int k);
void pupdate_T_postmult (Rhmatrix ppostmult_Tt[],
            Rhtranslink pTt[], int n);

//The next functions minimize the distance function according to
//the joint which is treated
double pF_objective (Rhmatrix Ti, Rhmatrix Th_i, double &qi, int i,
int k);
double pF_objective_without_ranks (Rhmatrix Ti, Rhmatrix Th_i,
double &qi, int i, int k);
double pF_calculate (Rhmatrix Ti, Rhmatrix Th_i, double &qi, int i,
 int k);
double pF_objective_arm (Rhmatrix *pTh[], Rhmatrix *pkpostmult_T[],
double **qi, int i);
double pF_objective_arm_without_ranks (Rhmatrix *pTh[],
Rhmatrix *pkpostmult_T[], double **qi, int i);
double pF_objective_joint_six (Rhmatrix *pTh[],
Rhmatrix *pkpostmult_T[], double **qi, int i);

//The next functions calculate the terms of the distance function
double pNUM_theta (Rhmatrix Ti, Rhmatrix Th_i, double salphai,
                double calphai, double ai);
double pDEN_theta (Rhmatrix Ti, Rhmatrix Th_i, double salphai,
                double calphai, double ai);
double ptindep (Rhmatrix Ti, Rhmatrix Th_i, double salphai,
                double calphai, double ai);
double pdm (Rhmatrix Ti, Rhmatrix Th_i, double salphai,
                double calphai);
double scale (int k);

//These methods are called by kineinverse_hand to calculate
//the inverse kinematics
```

```
    int   pkineinverse_iter (double *q [], Rhmatrix *pkppremult_T[],
          Rhtranslink *pkT[],Rhmatrix *pkpostmult_T[], Rhmatrix SA_hand[],
          int  iterations , double pepsilon );
    int   pkineinverse_iter_without_ranks (double *q [],
          Rhmatrix *pkppremult_T[], Rhtranslink *pkT[],
          Rhmatrix *pkpostmult_T[], Rhmatrix SA_hand[], int  iterations ,
          double pepsilon );
    int   pkineinverse_iter_hand_1 (double **q, Rhmatrix *pkppremult_T[],
          Rhtranslink *pkT[], Rhmatrix *pkpostmult_T[], Rhmatrix SA_hand[]);
    int   pkineinverse_iter_hand_2 (double **q, Rhmatrix *pkppremult_T[],
          Rhtranslink *pkT[], Rhmatrix *pkpostmult_T[], Rhmatrix SA_hand[],
          int  iterations , double pepsilon );
    double pdistance (Rhmatrix Ttcp, Rhmatrix Th);

    void setMessage(QString message);
    void printmatrix (Rhmatrix &matrix);
    void printmatrix (Rhmatrix &matrix, double scale );


protected:
    //These methods update the joint's values
    void pupdate_join (int join , double value , int k);
    void pupdate_join_arm (int join , double value );

public:
    //pS is the scale factor
    double *pS;
    //D-H parameters of each joint
    dh_parameters_joint2 **dh_parameters;

   // Value of the degrees of freedom of each chain
    int *dof;

   //Number of the kinematic chains
    int n_chain;
```

```
//Number of articulations that belong to different chains,
//but represent only one joint
int n_same;

//Number of all joints
int n_joint;

// Home values of the home positions
double **home;

// Transform Matrix of the world reference
Rhmatrix Tworld;

// Transform Matrix of the tools references
Rhmatrix *Ttool;

// Values of the current status of the robot's joins. The
// revolute joins in rads and the prismatic in mm
double ** status_join;

int error;
plug * list_plug;
double delta;

bool console_mode;
bool stop_kinematics;
QString message_out;
consoleWidget *console;

//These methods read the information of xml file
int init_dat (const char *argv);
int read_element_xml (QDomElement kine_element);

//This sends messages to the console
```

```
void send_stream(std :: ostringstream &oss);

void move_join (int join, double value, int k);
void move_inc_join (int join, double value, int k);
void mhome (int k);
void init (int pnewVal,int k);
void print_config (int n, int k);

//The next functions show the TCP value
void tcp (int k);
void tcp_arm (int k);
void show_tcp(int k);
Rhmatrix mat_tcp_arm(int k);
Rhmatrix mat_tcp(int k);

void show_status (int k);
void rank (int n, int k);
int kineinverse (Rhmatrix A_hand, double *q, int k);

//These functions calculate the inverse kinematics
int kineinverse_hand (Rhmatrix A_hand[], double **q);
int kineinverse_hand_retry (Rhmatrix A_hand[], double **q);

void settranform_Tworld (Rhmatrix T_world);
void settranform_Ttool (Rhmatrix T_tool, int k);

void move_path_join (double **qf, int movement);
void do_ready();

Rchain_hand ();
~Rchain_hand ();

// Write the home parameters and the kinematic Denavit-Hatenberg
// parameters of the joint i
void set_dh_pararmeters (int sigma, double theta, double alphai,
```

```
                    double ai, double di, double low_rank, double up_rank,
                    double maxspeed, double maxacc, int i, int k);

    void num_chains (int cadenes);

    void setconsole_mode(bool mode);

    void limits (int join, double &min, double &max, int k);

    QString name;

    void setconsole ( consoleWidget *console);

    public slots:
      void setValue(int i, double val);
      void slot_genpas_linear ();

    signals:
      void messageChanged(QString );

};
```

E.8 La resta de fitxers .h

La resta de fitxers .h que s'utilitzen en l'aplicació són els següents:

- new_geometric_object.h, new_grasping_object.h i new_kinematic_chain.h, que implementen les finestres que apareixen al seleccionar en l'aplicació un nou objecte geomètric, un nou objecte amb punts d'aprehensió o una nova cadena cinemàtica, respectivament.
- containers.h, que emmagatzema la informació que l'usuari ha introduït en les finestres anteriors.
- plug.h, que emmagatzema els valors actuals de cadascuna de les articulacions i els actualitza si aquests han canviat per un canvi en els sliders del panell de control o degut al

càlcul de la cinemàtica inversa.

- rhmatrix.h, rhrteuler i rhtranslink, que defineixen des de simples matrius fins a matrius de transformacions entre links.
- soqtcomposerotation.h i soqtcomposetranslation.h, que són una barreja de classes Qt i *engines* i s'encarreguen d'actualitzar les articulacions (de rotació o translació) del robot mostrat en l'escena.
- SoCoordinateAxis.h, que defineix els eixos d'un sistema de referència qualsevol.
- functions.h, que defineix diverses funcions matemàtiques usades en l'aplicació.