Controle de Braço Robótico via Interface Natural de Usuário com Sensor Kinect

Rafael F. R. Campos* Eduardo T. F. Santos**

* Departamento de Automação e Sistemas, Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA), BA, (e-mail: eduardot@ifba.edu.br).

** Departamento de Automação e Sistemas, Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA), BA, (e-mail: rafaelcampos@ifba.edu.br).

Abstract: Natural user interfaces (NUIs) are being used in the entertainment industry and beyond. This article detail the development of a didactic kit for applications in robotics, vision and control reusing an old robotic serial manipulator with DC motors. To design the controller, low-cost development boards (Arduinos and Shields) are used. The vision sensor used is the Kinect Sensor. The goal is perform the position control of the robot using gesture recognition, with the vision sensor providing the reference.

Resumo: As interfaces naturais de usuário (NUI's) estão sendo utilizadas na indústria de entretenimento e fora dela. Este trabalho detalha o desenvolvimento de um kit didático para aplicações em robótica, visão e controle reaproveitando um antigo braço robótico manipulador serial dotado de motores DC. Para projetar o controlador, são utilizadas placas de desenvolvimento de baixo custo (Arduinos e Shields). O sensor de visão utilizado é o Kinect. O objetivo é realizar o controle de posição do robô utilizando reconhecimento de gestos, com o sensor de visão fornecendo a referência.

Keywords: Robotics; Computer vision; Arduino; Kinect Sensor; Skeleton tracking

Palavras-chaves: Robótica; Visão Computacional; Arduino; Kinect; Skeleton tracking

1. INTRODUÇÃO

No campo da interação homem-máquina, interfaces naturais de usuário (NUIs) são interfaces que, de acordo com Steinberg (2012), são intuitivas, flexíveis e fluidas (o usuário por vezes não se dá conta de que está as utilizando). São frequentemente baseadas em paradigmas de interação do indivíduo com o mundo real, adaptáveis às preferências ou limitações de interação (objetivas ou subjetivas).

Um aspecto subjetivo das interfaces naturais de usuário é a ampliação dos sentidos vivenciada pelo seu uso. Como discorreu McLuhan (1974), quando o meio de interação com um artefato se torna tão simples e imediato a ponto de não ser mais percebido, o artefato transmuta-se em uma extensão de nós mesmos e forma em nós uma consciência não visual decorrente dessa interação. Isso eleva a experiência com o usuário de qualquer sistema ou produto a um patamar em que o artefato passa a ser parte de algum aspecto de sua personalidade.

O presente artigo propõe a construção de uma plataforma didática para desenvolvimento de "serious games"¹ baseada em sensor *Microsoft Sensor Kinect v2*, um dispositivo comercial NUI do console de videogame *Xbox One* para operar um braço robótico via gestos. Especificamente,

¹ Serious games são jogos eletrônicos com propósitos diversos que superam a indústria do entretenimento: Tratamento médico, treinamentos, simulações, visualização de dados. Barnes et al. (2009)

varrendo a posição das juntas do esqueleto do braço humano através da câmera de profundidade do dispositivo, estimando a posição das juntas no espaço, obtendo um modelo da posição desejada do braço robótico e realizando o controle de posição dos atuadores do robô através da plataforma Arduino.

2. CONCEITOS FUNDAMENTAIS

2.1 NUIs - Interfaces Naturais de Usuário

As primeiras interfaces de computação foram manipulação física dos circuitos, evoluindo para interface de linha de comando, *mouse*, ícones e *pads*. Na última década, houveram diversos avanços no sentido de reduzir a distância entre as metáforas da experiência humana com o mundo físico e a operação do *hardware* dos sistemas computacionais.

O reconhecimento de gestos é um desses avanços que se tornou popular após o abandono do paradigma de "toque à distância" (Goth, 2011, p.98), na década de 80. Em 2010, a Microsoft utilizou tal interação no produto *sensor Kinect*. Por causa do baixo preço de mercado e da disponibilidade de kits de desenvolvimento, se tornou plataforma de desenvolvimento na engenharia, computação, conforme apontou Zhang (2012), arte e na cultura *maker*².

² Os aspectos mais marcantes da cultura *maker* estão descritos em Hatch (2014).

Fig. 1. Sensor Microsoft Kinect v2

De acordo com (Szeliski, 2010, p. 3), o problema da visão computacional é o desenvolvimento de técnicas para interpretação de uma imagem em diversos níveis cognitivos. Assim, não é um campo de estudos com um conjunto de técnicas bem definido; a aplicação final envolve técnicas tão variadas (Corke, 2017, p.286) quanto estudos da física da formação e captura da luz, processamento de imagens e algoritmos para extração das características desejadas em seus diversos paradigmas.

De acordo com Corke (2017), um robô é uma “máquina orientada à objetivos que pode sentir, planejar e agir”. São comumente classificados como fixos e móveis, e dentre os fixos, um tipo particular interessa à este trabalho: o braço robótico do tipo manipulador serial. Ele consiste de elos unidos por juntas acionadas por motores.

3. ARQUITETURA DO SISTEMA

3.1 O sensor Kinect

Hack é o resultado inteligente de algum agente que aplica seu engenho para conseguir algum resultado, como apontou Barbosa (2015). Foi com um *hack* que o *sensor Kinect* deixou de ser mais um acessório de jogo para ser intensamente explorado por artistas, *makers* e cientistas. As primeiras bibliotecas de suporte à programação do dispositivo foram desenvolvidas por programadores independentes. O primeiro trabalho relevante então é a biblioteca *libfreenect*, disponível em *OpenKinect* (2018).

No mesmo ano, a PrimeSense lança os *frameworks* *OpenNI* e *NITE*, que fornecem um *driver* e uma camada de abstração, com acesso à reconhecimento de voz, rastreamento de corpo e reconhecimento de gestos. Várias obras seguem para o público *maker* a partir deste ponto. Borenstein (2012) publicou um guia para o público que pretendia utilizar o *sensor Kinect* em seus projetos.

Ainda na esteira dos desenvolvedores independentes de 2010, surgem trabalhos publicados na academia, por exemplo, mapeamento de ambientes 3D em Perez Bonnal (2010) e rastreamento de *skeleton* em Kar et al. (2010). A Microsoft então libera em 2011 *drivers* oficiais. Em 2013, foi lançado pela Microsoft o *Kinect v2* (Fig 1), dispositivo adotado para reconhecimento de gestos neste artigo.

3.2 Visão Computacional

A técnica utilizada neste trabalho para reconhecimento de gestos é o rastreamento de esqueleto (*skeleton*). O *skeleton* é uma estrutura de dado do tipo grafo, cujos elos são os “ossos” e os nós são as “juntas”. A topologia do grafo é

invariante, e as juntas contêm a informação de um vetor que representa a posição da junta no espaço tridimensional, considerando a origem do sistema de coordenadas a posição do sensor.

A extração da estrutura de dado, desde a imagem real até o *skeleton*, é realizada em duas etapas, a primeira é a reconstrução da superfície e a segunda, através técnicas de *machine learning*.

A tecnologia de sensoreamento de profundidade do sensor *kinect 2* é baseada cálculo de deslocamento de fase do pulso infravermelho refletido Lun and Zhao (2015).

A imagem é então segmentada e aplicada ao algoritmo de aprendizado de máquina *Random Forest*. Os detalhes da etapa de treinamento, assim como seus desafios computacionais podem ser encontrados em Shotton et al. (2011). Após a obtenção da nuvem de pontos, são determinados os centros de massa para obter o ponto mais provável da junta no espaço 3D.

O *hardware* então fornece um *stream* de dados, que é um fluxo de matrizes que informam a posição das juntas do *skeleton*, e outros metadados necessários, que são tratados pelo *software* desenvolvido neste trabalho.

3.3 Planta

A planta a ser controlada é classificada como um robô manipulador serial de 4 eixos, do tipo de coordenadas de revolução, com pequenas dimensões e peso de aproximadamente 7 kg, dotado de balancins na junta 2 e molas em sua contraposição, sendo utilizado em bancada, conforme mostra a Fig. 2. Foi comercializado pela empresa Didacta Italia e possui uma ferramenta final tipo garra acionada pelo mesmo motor DC que aciona os eixos, com uma chave fim de curso. Os parâmetros Denavit-Hartenberg foram levantados por (Pires and Maravilha, 2016, p. 54).

Fig. 2. Robô manipulador serial RD5 NT e modelo

Os acionadores são compostos por um motor DC, produzido pela Maxon (2013), cujas características principais são:

- Valores nominais: tensão 12V, potência 2.5 W, rotação 3050 rpm, torque contínuo máximo 3.39 mNm, corrente contínua máxima 204 mA
- Eficiência máxima de 71% na tensão de operação de 12 V
- Corrente e rotação a vazio: 10.3 mA, 6480 rpm
- Torque resistente que causa bloqueio do rotor 5.25 mNm
- Corrente de partida 376 mA
- Ganho velocidade/tensão 555 rpm/V

Junta	Mínimo	Máximo
Junta 1	0.464 V	4.434 V
Junta 2	0.374 V	4.364 V
Junta 3	1.837 V	3.150 V
Junta 4	0.502 V	4.494V

Tabela 1.

O motor é acoplado a dois conjuntos mecânicos de redução de velocidade, de relação total de 1:500. A medição de posição do eixo é feita por transdutores rotativos de precisão, do tipo resistivo, com torque resistente durante o movimento de 0.5 Ncm máx. e torque resistente no início do movimento de 50 Ncm e pode ser consultado no catálogo Sfernice (2015) com o código ECO78CSA502W6817. São potenciômetros de 5 k Ω . O mapeamento dos limites físicos dos eixos e tensões aplicando 5V nos terminais do transdutor estão representado na Tab. 1.

4. INTEGRAÇÃO DO SISTEMA

4.1 Hardware

Neste trabalho, o sistema completo é um manipulador serial acionado por pontes H e as juntas são motores de corrente contínua (Motores CC), conforme o diagrama de blocos da Fig. 3.

Fig. 3. Diagramas de blocos do sistema de controle

A versão de placa controladora escolhida foi o Arduino Mega (Fig.4), por causa da disponibilidade de portas analógicas acessíveis sem necessidade de solda, mesmo com *shield* instalado, porte do compilador *GNU C Compiler* (gcc) para plataforma AVR, ampla disponibilidade de bibliotecas de código aberto, facilidade de programação e baixo custo, facilitando a replicação por parte dos estudantes.

Fig. 4. Placa microcontrolada Arduino Mega

O microcontrolador que o Arduino Mega porta é o ATmega2560. É um circuito integrado CMOS, de arquitetura RISC, com 135 instruções. Possui 4Kbytes de EEPROM,

8kBytes de SRAM, e um oscilador de cristal de 16 MHz. Atmel (2005).

O driver selecionado foi o *Adafruit Motor Driver v1*, documentado por Fried (2018)(Fig. 5) devido ao baixo custo, disponibilidade de biblioteca de software e esquemáticos em Adafruit (2011), volume de placas *clone* acessíveis no mercado local e grande comunidade de usuários.

O *driver* possui suporte para servomotores, motores de passo e motores DC. Possui suporte para até 4 motores, e pode reverter a posição utilizando circuitos integrados L293D, que implementam circuitos de ponte H. O *driver* fornece 0.6A por *bridge* (1.2A de pico) com proteção contra desligamento térmico, e opera com motores que de 4.5V a 25V. Para proteger o circuito na partida, existem resistores *pull-down* que mantêm os motores desativados durante a energização Fried (2018).

Fig. 5. Driver para controle de motores da Adafruit

4.2 Obtenção dos ângulos do braço humano com o sensor Kinect

Foi utilizado neste trabalho o ambiente MATLAB 2018b, e as toolboxes Team (2018) de processamento e aquisição de imagens. Primeiro, foi realizada a comunicação da câmera com o ambiente MATLAB para obtenção do fluxo de vídeo e dos metadados. O posicionamento ótimo para aquisição foi obtido com o sensor a 60 cm do piso, e o corpo a 2 m do sensor. As especificações necessárias de *hardware* para o ambiente de aquisição utilizadas foram:

- Conexão USB 3.0 para que o fluxo de vídeo seja estabelecido.
- Processador 7ª geração Intel Core i7, 4 núcleos, 2.70 GHz, 14nm
- 8 Gb de RAM
- Placa gráfica Intel 620, 14nm, 1150 MHz

Após o estabelecimento de *stream* de vídeo e metadados, é iniciada a tentativa de obter um *skeleton* da cena. O código 1 realiza essa tentativa através dos seguintes passos:

- o programa monitora o fluxo de metadados e aguarda até que, pelo menos, exista *skeleton* em um *frame* do *stream*.
- Assim que há um ou mais *skeletons* na cena, captura as informações das juntas do primeiro *skeleton* rastreado da cena e armazena.
- Calcula os ângulos das juntas.
- Desliga a câmera

Listing 1: Algoritmo de aquisição do *skeleton*

```
i = 0;  
while 1  
    disp('Esperando frames')  
    while get(vid, 'FramesAvailable') < 1
```



```

        unavailable=1;
    end
    disp('Procurando')
    [data, time, metaData] = getdata(vid,10);
    existeEsqueleto = metaData.IsBodyTracked;
    if any(existeEsqueleto(:))
        for j = 1:6
            if existeEsqueleto(1,j)
                flagEsq = j;
                break;
            end
        end
        for k = 1:10
            if metaData(k).IsBodyTracked(1,flagEsq)
                flagFrame = k;
                break;
            end
        end
        mao =
            metaData(flagFrame).JointPositions(12,:,flagEsq);
        pulso =
            metaData(flagFrame).JointPositions(11,:,flagEsq);
        cotovelo =
            metaData(flagFrame).JointPositions(10,:,flagEsq);
        ombro =
            metaData(flagFrame).JointPositions(9,:,flagEsq);
        centroOmbro =
            metaData(flagFrame).JointPositions(3,:,flagEsq);
        coluna =
            metaData(flagFrame).JointPositions(2,:,flagEsq);
        referencialCamera = [0,0,0]
        angulo1 = calculoAngulo(pulso, mao, pulso, cotovelo)
        angulo2 = calculoAngulo(pulso, cotovelo, ombro,
                                cotovelo)
        angulo3 = calculoAngulo(ombro, cotovelo,
                                centroOmbro, coluna)
        angulo4 = calculoAngulo(centroOmbro,
                                referencialCamera, ombro, cotovelo)
        break;
    end
    i = i+1
end
stopCamera
    
```

Um diagrama de como as juntas são mapeadas estão exibidas na Fig. 6, e as posições 3D no espaço são utilizadas para determinar os ângulos entre os membros.

O robô RD5 NT só possui 4 graus de liberdade, logo a pose do braço humano não é realizável. Portanto, foram selecionados os seguintes ângulos do braço humano para reprodução, conforme Fig. 6:

- Pulso direito, no espaço [11]
- Cotovelo direito, no espaço [10]
- Ângulo na vertical entre a coluna vertebral (semireta: coluna [2] e centro do ombro [3]) e o braço (semireta: ombro [9] e cotovelo [10])
- Ângulo na horizontal entre a coluna vertebral (semireta: origem do sistema de coordenadas e centro do ombro [9]) e o braço (semireta: ombro [9] e cotovelo [10])

4.3 Comunicação do Ambiente MATLAB com o controlador embarcado

Foi criado um simples protocolo para comunicação entre o ambiente do MATLAB e o Arduino para envio dos

Fig. 6. Identificação das juntas do *skeleton* e os índices do metadado. Imagem retirada de Ahmed and Sabir (2017).

set-points para o controlador do robô, utilizando uma comunicação serial através da porta virtual COM do PC. Segue a descrição:

- São enviados somente um conjunto de *set-points* por vez. O MATLAB abre a comunicação com a porta serial e permanece em escuta para verificar se há algum pedido de dados.
- O Arduino, por sua vez, após a inicialização, abre a serial e começa a enviar a solicitação para enviar dados.
- Quando o MATLAB recebe a solicitação, ele envia uma *string* de 4 dados de *set-points* concatenados para a porta serial.
- Agora, o Arduino permanece em escuta, e quando há 12 bytes disponíveis na porta serial (4 *set-points* de 3 bytes cada um), ele os lê e armazena.
- Após isso, o Arduino envia a informação de que o dado já foi lido. O MATLAB, por sua vez, permanece em escuta de confirmação.
- Quando a confirmação acontece, ele fecha a conexão serial.
- Os *set-poits* são recebidos como variáveis *strings*. É utilizada a biblioteca padrão C para transformar esses valores em numéricos, com o fim de ser processados pelo Arduino.

4.4 Desenvolvimento da estratégia de controle PID dos eixos

O algoritmo de controle PID dos eixos utiliza a biblioteca Beauregard (2012). Essas são as características do algoritmo implementado:

- A taxa de amostragem foi configurada em 200ms.
- A quantização do sensor foi configurada em 10bits, e a saída para o acionador, em 7 bits.
- Existe também um tratamento com relação ao fenômeno de *wind-up*, fixando “batentes” de sinal, valores máximo e mínimo que podem ser enviados como PWM e no termo integral, evitando tanto a saturação do controlador quanto do atuador.

- O controlador também lida com o problema de alto atrito estático das juntas, implementando um valor mínimo que é enviado pelo controlador para o *driver*.
- É estabelecido um pequeno valor de histeresse para evitar a oscilação contínua no caso de eixos que necessitem de altos ciclos de trabalho para vencer o atrito estático.

No código 2, está o algoritmo da estratégia de controle dos eixos. somente os valores das constantes e endereços de entrada e saída são diferentes para cada eixo.

Listing 2: Algoritmo de controle dos motores

```
#include <AFMotor.h>
#include <PID_v1.h>
const batente_min = 30;
const histeresse = 5;
AF_DCMotor motor(1);
int sensor = A8;
double angulo;
double pwmreal = 0;
int pwm;
double setpoint = 180;
PID eixo(&angulo, &pwmreal, &setpoint, 1, 1, 1, DIRECT);
void setup() {
    eixo.SetMode(AUTOMATIC);
    eixo.SetSampleTime(200);
}
void loop() {
    angulo = map(analogRead(sensor), 0, 1023, 0, 360);
    eixo.Compute();
    pwm = (int)round(pwmreal);
    if (pwm < batente_min)
        pwm = batente_min;
    if (angulo < setpoint-histeresse){
        motor.run(FORWARD);
    }
    else if (angulo > setpoint+histeresse) {
        motor.run(BACKWARD);
    }
    else{
        motor.run(RELEASE);
    }
    motor.setSpeed(pwm);
}
```

5. RESULTADOS

Para avaliar cada eixo, foi aplicado uma entrada de tensão PWM, em uma escala de 0 a 100% (δ) de ciclo de trabalho e realizada a medição da saída o θ (ângulo em graus). Como os motores são DC, é esperada uma função linear, cujo coeficiente angular é a taxa ω/δ , ou o ganho de velocidade.

Para realizar a identificação do ganho, foi aplicada a menor entrada degrau possível para vencer o atrito estático da junta.

Os gráficos de aquisição para o movimento progressivo podem ser vistos nas Fig. 8 e 9. No eixo das ordenadas, está o valor do ângulo em graus, e no eixo das abscissas, cada unidade de tempo corresponde a 50 ms.

Para cada eixo, foi considerado o pior caso de ganho de velocidade, seja o movimento progressivo ou regressivo. Os resultados normalizados e o ganho mínimo estão relacionados na Tab. 2.

Fig. 7. Deslocamento dos eixos para medição da velocidade angular após aplicação de um degrau de *duty cycle* (δ) no modulador PWM, movimento progressivo dos eixos

Fig. 8. Deslocamento dos eixos, movimento regressivo

Eixo	ω/δ	δ_{min}
1	0.1723	30%
2	0.0057	60%
3	0.0163	100%
4	0.0076	60%

Tabela 2.

Existem algumas variações bruscas, devido à falhas nos potenciômetros (desgaste ou contaminação do material resistivo). Tais variações podem ser minimizadas utilizando um filtro FIR passa-baixas.

O software de detecção de *skeletons* enviava para o *workspace* 20 frames por segundo e 100% de *skeletons* da cena rastreados, com tempo máximo de rastreamento 20 segundos.

Os dispositivos mecânicos mostraram dificuldades para modelagem e controle dinâmico. Existe atrito acima do esperado devido à idade e condições de armazenamento anteriores do braço robótico. Mesmo assim, a aplicação de torque mínimo nos momentos de reversão de velocidade garantiram o desempenho para alcance de posição desejada. A velocidade máxima dos eixos, entretanto, não permite o rastreamento de trajetórias.

O alto atrito torna impraticável a aplicação de uma estratégia PID propriamente dita no eixo 3. O algoritmo nesse caso, vai operar como uma chave liga-desliga. No caso dos eixos 2 e 4, é esperada uma oscilação do controlador, por causa do alta energia necessária para o atuador e baixo ganho de velocidade da junta. Um controle PID satisfatório só é esperado para o eixo 1.

6. CONCLUSÃO

O trabalho demonstra a utilização de equipamentos de baixo custo e um apanhado de detalhes importantes do desenvolvimento a fim de disponibilizar uma plataforma didática para estudantes de graduação se envolverem em projetos de *serious games*. O uso de NUIs com *serious games* é natural e sugere uma infinidade de aplicações em serviços e produtos.

Aplicações atuais em terapia e reabilitação médica utilizando dispositivos como Microsoft Kinect, processadores digitais de imagem, visão computacional e robótica assistiva são demonstrados em todo o mundo com alto potencial de geração de valor, como pode ser visto em Chang et al. (2011), Mousavi Hondori and Khademi (2014).

Outras aplicações possíveis são a manipulação de materiais em ambientes não estruturados, como locais em que há presença ou risco de presença de agentes químicos, biológicos ou físicos perigosos, como radiações ionizantes ou substâncias danosas ao corpo humano em baixíssimas concentrações.

A bancada didática pode ser melhorada ao utilizar módulos de câmeras de baixo custo ou utilizar a experiência de interação com *speech recognition*.

REFERÊNCIAS

- Adafruit (2011). Adafruit motor shield for arduino. <https://github.com/adafruit/Adafruit-Motor-Shield-for-Arduino>.
- Ahmed, M.H. and Sabir, A.T. (2017). Human gender classification based on gait features using kinect sensor. In *Cybernetics (CYBCONF)*, 2017 3rd IEEE International Conference on, 1–5. IEEE.
- Atmel (2005). Atmega2560 datasheet.
- Barbosa, D. (2015). Hackers: Subvertendo sistemas. *Facta*, (3), 17.
- Barnes, T., Encarnação, L.M., and Shaw, C.D. (2009). Serious games. *IEEE Computer Graphics and Applications*, 29(2), 18–19.
- Beauregard, B. (2012). Improving the beginner's pid. *Web*. <http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/>.
- Borenstein, G. (2012). *Making things see: 3D vision with kinect, processing, Arduino, and MakerBot*. "O'Reilly Media, Inc."
- Chang, Y.J., Chen, S.F., and Huang, J.D. (2011). A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities. *Research in developmental disabilities*, 32(6), 2566–2570.
- Corke, P. (2017). *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised*, volume 118. Springer.
- Fried, L. (2018). Adafruit motor shield. URL <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield.pdf>.
- Goth, G. (2011). Brave nui world. *Communications of the ACM*, 54(12), 14–16.
- Hatch, M. (2014). *The maker movement manifesto: Rules for innovation in the new world of crafters, hackers, and tinkerers*. McGraw-Hill Education New York.
- Kar, A. et al. (2010). Skeletal tracking using microsoft kinect. *Methodology*, 1(1), 11.
- Lun, R. and Zhao, W. (2015). A survey of applications and human motion recognition with microsoft kinect. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05), 1555008.
- Maxon (2013). *F2130 30 mm, mm, Precious Metal Brushes 2.5 CLL 2.5 Watt*.
- McLuhan, M. (1974). *Os meios de comunicação: como extensões do homem*. Editora Cultrix.
- Mousavi Hondori, H. and Khademi, M. (2014). A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. *Journal of medical engineering*, 2014.
- OpenKinect (2018). libfreenect. <https://github.com/OpenKinect/libfreenect>.
- Perez Bonnal, E. (2010). 3d mapping of indoor environments using rgb-d kinect camera for robotic mobile application.
- Pires, O.P. and Maravilha, P.H.V. (2016). Controle pid de posição do manipulador robótico rd5 nt por ganho agendado.
- Sfernice, V. (2015). Precision rotative transducers, conductive plastic, economic series (eco). URL <http://elcodis.com/parts/3005888/EC078CSA502W6817.html#datasheet>.
- Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR)*, 2011 IEEE Conference on, 1297–1304. Ieee.
- Steinberg, G. (2012). Natural user interfaces. In *ACM SIGCHI conference on human factors in computing systems*. URL https://www.cs.auckland.ac.nz/compsci705s1c/exams/SeminarReports/natural_user_interfaces_gste097.pdf.
- Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- Team, M.I.A.T. (2018). Matlab image acquisition toolbox package for kinect for windows sensor. The MathWorks, Natick, MA, USA.
- Zhang, Z. (2012). Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2), 4–10.