DESENVOLVIMENTO DE UM SISTEMA PARA INTEGRAÇÃO DE CÉLULAS ROBÓTICAS INDUSTRIAIS COM FOCO NA INDUSTRIA 4.0

Marcos Paulo Torre*†, Gabriel Carvalho Garcia*†, Sander Soares Queiroz‡, Myriam Souza de Oliveira Dias‡, Marcone Jamilson Freitas Souza§, Gustavo Medeiros Freitas*

*Instituto Tecnológico Vale - Ouro Preto, MG, Brasil

†Escola de Minas - Universidade Federal de Ouro Preto - Ouro Preto, MG, Brasil

‡Vale S.A. - Engenharia Mineral e Laboratórios - Corredor Sudeste - Nova Lima, MG, Brasil

§Departamento de Computação - Universidade Federal de Ouro Preto - Ouro Preto, MG, Brasil

Emails: marcos.torre@itv.org, gabrcg@gmail.com, sander.queiroz@vale.com, myriam.souza.dias@vale.com, marcone@iceb.ufop.br, gustavo.medeiros.freitas@itv.org

Abstract— This work aims to develop a system for industrial robotic cells integration, focusing on Industry 4.0, using softPLC and 3D perception approaches. Most industrial robots use proprietary software and closed architecture drivers, and as a result, the integration with devices from different manufacturers becomes inflexible. Thus, we propose the development of an open-source tool to simplify the system development and integration using the framework Robot Operating System (ROS). The work's development aims the application in a robotic cell for iron ore samples manipulation. As tests in industrial environments demand the stop of machines directly involved in the production process, laboratory tests were done using equipment with the same characteristics as those used in the real process, such as an ABB IRB 120 robotic arm, in conjunction with a Festo didactic plant, capable of distributing, separating and sorting parts. Using a depth camera, we have programmed the robotic manipulator to perform pick and place tasks, identifying the parts position and inserting them into the circuit. Subsequently, the parts follow simulated processing routes according to the cell processes.

Keywords— System Integration, Industry 4.0, 3D Perception, softPLC, Industrial manipulators.

Resumo— O presente trabalho propõe o desenvolvimento de um sistema, com foco na Industria 4.0, para integração de células robóticas industriais, utilizando abordagens de *softPLC* e percepção 3D. A maioria dos robôs industriais utilizam *softwares* proprietários e controladores de arquitetura fechada, tornando inflexível a integração com dispositivos de fabricantes diferentes. Dessa forma, é proposto o desenvolvimento de uma ferramenta de código aberto com o objetivo de simplificar a integração, expansão e inclusão de algoritmos de alto nível nos sistemas, utilizando o *framework Robot Operating System* (ROS). O trabalho foi realizado visando a aplicação em uma célula robótica para manipulação de amostras de minério de ferro. Como a realização de testes em ambientes industriais demanda a parada de equipamentos diretamente envolvidos na produção, os testes foram realizados em equipamentos com as mesmas características dos utilizados nos processos reais, como um braço robótico ABB IRB 120 e uma planta didática Festo para separação, distribuição e classificação de peças. Com o auxílio de uma câmera de profundidade, o braço robótico foi programado para realização de tarefas de *pick and place*, identificando a posição das peças e inserindo-as no circuito. Posteriormente, as peças seguem por rotas de processamento simuladas de acordo com os processos da célula.

Palavras-chave— Integração de Sistemas, Industria 4.0, Percepção 3D, softPLC, Manipuladores industriais.

1 Introdução

A capacidade de efetuar tarefas complexas e repetitivas com alto grau de precisão, velocidade e repetibilidade, tornaram os braços robóticos industriais um importante aliado no caminho para a competitividade nos setores industriais. Além do mais, estes podem ser aplicados em ambientes insalubres e de risco sem perda de desempenho, sendo uma importante ferramenta para evitar danos à vida dos colaboradores.

Mesmo com as inúmeras vantagens de sua utilização, os braços robóticos industriais são geralmente programados para efetuar tarefas pré-programadas, sem interação com o ambiente (Moriano Martín, 2013). Esses dispositivos utilizam softwares proprietários e controladores de arquitetura fechada, dificultando a integração com dispositivos de fabricantes diferentes.

De modo geral, células robóticas não são compostas unicamente por robôs, sendo que as etapas do processo são executadas por equipamentos diferentes, podendo existir controladores que se comunicam em protocolos distintos, resultando em ilhas de automação que necessitam de integração.

Segundo Bartolomeu et al. (2005), um dos grandes desafios da robótica é justamente como integrar as informações vindas de todos os algoritmos e sensores presentes em um processo, de modo a gerar comandos e controlar os diferentes dispositivos de atuação do robô, garantindo que a tarefa seja executada de modo correto, sem colocar em risco o robô e aqueles que o cercam.

Os fatos previamente descritos foram vivenciados em uma célula robótica de um laboratório físico da empresa Vale S.A.. Os equipamentos que compõem a célula foram adquiridos de forma gradual, e as mudanças no cenário da indústria da mineração impossibilitaram a contratação de serviços para integração das máquinas. Cada uma dessas possui seu PLC, do inglês *Programmable Logic Controller*, sem comunicação com os demais.

Para possibilitar troca de dados e sincronismo de tarefas, é necessário implementar a comunicação entre os controladores de cada dispositivo.

Para a aplicação de ferramentas alinhadas com as premissas da Industria 4.0, é importante que o sistema esteja preparado para tal desde sua concepção, o que não é alcançado facilmente por meio de técnicas de integração convencionais, considerando a variedade de padrões existentes e inflexibilidade de sistemas proprietários.

Este trabalho mostra uma abordagem com foco na Industria 4.0 para integração dos equipamentos da célula robótica estudada, no qual é proposto um sistema flexível, de código aberto e capaz de se comunicar com alguns dos principais controladores, robôs e redes de campo utilizados na industria, tais como CanOpen, Ethernet/IP, Profinet e Modbus TCP. A plataforma tem o objetivo de padronizar e facilitar o cruzamento de informações provenientes de sensores, algoritmos e tarefas, para tomada de decisões de uma maneira mais eficaz.

O desenvolvimento da plataforma foi baseado no *framework Robot Operating System* (ROS) e sua extensão ROS-Industrial, que são ferramentas de código aberto criadas para incentivar a implementação colaborativa de *softwares* para sistemas robóticos.

O sistema também será capaz de interagir com o espaço de trabalho por meio de percepção 3D, de forma que o ambiente ao redor possa ser compreendido e as trajetórias do braço robóticos sejam efetuadas baseadas nessa percepção, sendo uma alternativa às rotas pré-programadas e fixas.

Devido as dificuldades de realização de testes em ambientes industriais, que demandam parada de equipamentos envolvidos diretamente na produção, testes de conceito foram efetuados em laboratório, por meio de equipamentos industriais representativos, com o objetivo de validar seu uso nas próximas etapas de integração da célula. Ainda, segundo Zuehlke (2010), propostas que visam substituir tecnologias bem estabelecidas devem ser inicialmente testadas fora do ambiente industrial, e aplicadas somente quando forem garantidas sua robustez, segurança e integridade.

Dessa forma, o presente trabalho visa fornecer uma opção de futuro para integração de células robóticas industriais, onde diversos algoritmos de alto nível para interação com o sistema possam ser aplicados de uma maneira mais simples e direta, como os de otimização, tratamento de imagens e visão computacional.

2 Automação do Laboratório Físico

A célula robótica do laboratório físico da Vale tem a função de realizar transformações físicas nas amostras de minério de ferro para preparação de pastilhas fundidas, que posteriormente serão analisadas quimicamente e fornecerão informações importantes para o controle de qualidade do produto, como teor de ferro, contaminantes, entre outros.

O projeto do laboratório automatizado tem o objetivo de integrar esteiras de distribuição e de saída, uma estufa, um britador Herzog HP-CS/A, dois moinhos pulverizadores Herzog HP-M1500, uma unidade dosadora Herzog HP-SCD e um manipulador industrial ABB IRB 4600. Um esquema desse projeto está ilustrado na Figura 1. As principais metas são a redução do erro de amostragem, redução de trabalho manual e aumento de performance.

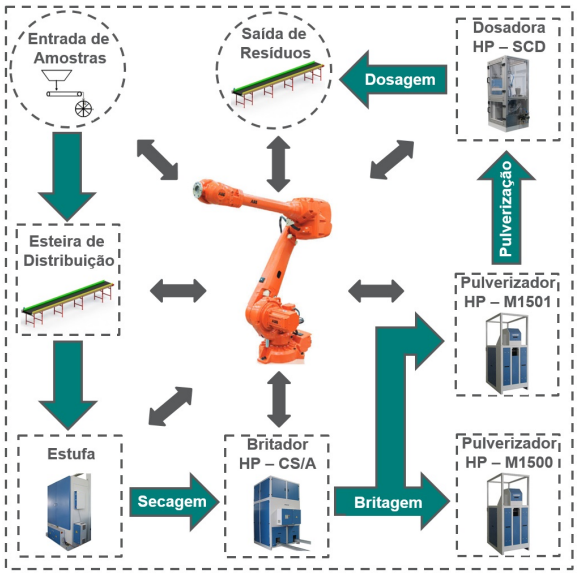


Figura 1: Layout do laboratório físico

O processo começa com o operador despejando a amostra em um copo metálico e posicionando-o na esteira de distribuição. Posteriormente, o manipulador industrial tem a função de realizar tarefas de *pick and place*, inserindo e retirando as amostras dos equipamentos, respeitando o fluxo mostrado na Figura 1.

Geralmente, um PLC geral não vinculado a nenhuma máquina da célula é inserido como elemento de interface e centralizador de dados. Esta abordagem demanda a inclusão de equipamentos que normalmente possuem um alto custo associado, muitas vezes desempenhando apenas a função de comunicação, visto que cada equipamento já possui um programa interno e apenas o interfaceamento de comandos é necessário.

Em outros casos, alguns controladores que já fazem parte do processo e que possuam as interfaces desejadas são utilizados para desempenhar essa função, reservando parte de seu processamento para executar a tarefa de comunicação. Essa arquitetura torna confusos os mapas de comunicação da rede, acarretando na dificuldade de expansão e manutenção do sistema, além da possibilidade de sobrecarga do controlador, dependendo do processo em execução.

A implementação de algoritmos de alto nível, como para tratamento de imagens e percepção 3D, demandam um processamento computacional não disponível em PLCs industriais convencionais. Logo, esses algoritmos são normalmente executados em computadores ou servidores, sendo posteriormente interfaceados com os devidos controladores industriais. Com base nessas informações, este trabalho propõe que o mesmo ambiente computacional efetue a função de PLC central e execute os algoritmos de alto nível responsáveis por tarefas mais complexas.

3 Revisão bibliográfica

A revisão bibliográfica foi baseada em aplicações alternativas de integração de equipamentos voltadas para a indústria. O conceito da plataforma proposta se aproxima bastante da abordagem de *softPLC*, que se baseia no controle de equipamentos por meio de computadores pessoais, envolvendo ganhos de flexibilidade, baixo custo e maior capacidade de processamento. Como sistemas industriais exigem robustez, mesclas entre sistemas baseados em *softPLC* e controladores tradicionais também devem ser consideradas.

Visando demonstrar a flexibilidade da aplicação proposta e capacidade de interação com o mundo ao redor da célula robótica, foram revisadas algumas técnicas para tratamento de mapas de profundidade, que posteriormente serão aplicadas para identificação da posição de objetos.

3.1 Técnicas de Integração

Como execução prática em equipamentos industriais, Martinov et al. (2016) propõem a implementação de um sistema de controle automático com *hardware* independente para tornos e fresadoras CNC. A solução contempla o desenvolvimento de lógicas de controle em controladores lógicos programáveis baseados em *software* (*softPLC*), permitindo a unificação dos algoritmos para comando de equipamentos auxiliares envolvidos no interfaceamento de máquinas. Essa abordagem permite a divisão dos níveis de implementação do programa de controle e também do *hardware* do PLC.

Por meio da configuração direta no ambiente de programação, a solução apresenta flexibilidade de adaptação entre diferentes grupos de módulos de I/O e dispositivos auxiliares. O artigo também propõe um modelo de configuração de uma área de memória compartilhada para acesso dos dados provenientes dos módulos de *hardware* disponíveis. Essa área de memória é composta por uma matriz, onde os dados físicos são lidos, sendo esta matriz preenchida de acordo com o *ID* e *Vendor ID* de cada elemento. Assim, os dados lógicos podem ser acessados de uma maneira estruturada.

Visando apresentar as vantagens e desvantagens da implementação de um sistema de controle unificado, outro artigo de Martinov et al. (2017) enfoca o controle de máquinas periféricas utilizando a mesma abordagem de *softPLC*. A implementação do sistema centralizado em uma única CPU apresenta algumas desvantagens. No caso da ocorrência de um erro crítico de *hardware*, todas as tarefas serão interrompidas, o que acarretará em consequências imprevisíveis. Dessa forma, a abordagem contempla, além de uma placa contendo o controlador baseado em *software*, outro controlador que garante as funções de segurança em caso de falha do sistema central.

A adoção de controladores auxiliares responsáveis pelas tarefas principais das máquinas garante a distribuição da carga computacional das tarefas de controle em múltiplos processadores e o aumento da confiabilidade do sistema, visto que o gerenciamento, controle das atividades, funções de monitoramento e diagnósticos dos módulos de *hardware* estarão distribuídos.

No estudo realizado por Nocoń and Choiński (2006) os autores propõem um sistema de controle tolerante a falhas, que visa aumentar a flexibilidade de programação para realização dos procedimentos experimentais de uma planta piloto para tratamento de água. Todos os sensores e atuadores do sistema são conectados em PLCs industriais em sua forma padrão, porém, o programa principal é executado em um computador.

Como um computador pessoal não possui a robustez de um controlador industrial, o PLC também é programado para verificar o status da comunicação com o computador e, em caso de falhas, o PLC assume o controle da planta. Dessa maneira, os controladores industriais foram utilizados como dispositivos de entrada e saída pelo computador, além de possuir um programa *hot-standby* que monitora a execução do processo.

3.2 Percepção 3D

A percepção 3D é uma importante ferramenta para que sistemas robóticos, que trabalham em ambientes não estruturados ou que possam sofrer alterações de *layout*, possam interagir com o mundo a sua volta (Rusu and Cousins, 2011). Sensores com essa capacidade de percepção são aptos a representar ambientes em nuvem de pontos. Em cada ponto da nuvem gerada estão associados sua posição no espaço tridimensional e também sua cor em RGB. Essas características possibilitam a extração de informações importantes do ambiente, como a posição e orientação de objetos, utilizando técnicas conhecidas por pré-processamento e segmentação.

3.2.1 Pré-processamento

Consiste na preparação da nuvem de pontos para as etapas de segmentação. As etapas principais do pré-processamento são divididas em:

- *Voxel Grid Downsampling:* A aplicação deste filtro cria uma malha de pequenas caixas 3D ao longo da estrutura da nuvem de pontos. Posteriormente, todos os pontos situados dentro da caixa são reduzidos ao seu centroide (WIL, 2018), reduzindo a densidade do mapa de profundidade.

- *Filtro PassThrough:* O Filtro *PassThrough* limita as dimensões da nuvem de pontos de entrada. Em outras palavras, é possível eliminar os pontos que se encontram fora de um determinado intervalo do eixo tridimensional.

3.2.2 Segmentação

Com a região limitada, são aplicadas técnicas de identificação de formas conhecidas. Com os objetos extraídos, aplicam-se técnicas clusterização para que as peças sejam isoladas e tratadas individualmente.

- *RANSAC:* Segundo Derpanis (2010), o *RANdom SAmple Consensus*, é um algoritmo para levantamento de modelos matemáticos robustos a partir de dados que apresentem *outliers*. Dessa forma, a técnica pode ser aplicada para identificação de geometrias conhecidas, como planos, cilindros e esferas, presentes na nuvem de pontos. Logo, é fornecido o modelo matemático da forma desejada e dois subconjuntos são gerados, o que contém (*inliers*) e o que não contém (*outliers*) a geometria desejada. De acordo com a aplicação, o mapa de profundidade pode ser manipulado a fim de manter apenas os *inliers* ou somente os *outliers*. Ainda é possível manter um intervalo contaminado com *outliers*, que pode ser manipulado de maneira que represente o modelo da melhor maneira.

- *DBSCAN:* Segundo Zhou et al. (2000), o *Density-Based Clustering Methods* é um algoritmo capaz de encontrar regiões mais densas (*clusters*) nos dados de entrada, que podem ser medidas pelo número de objetos próximos a um determinado ponto. Assim, o DBSCAN é capaz de identificar *clusters* de formas variadas e obter bom desempenho lidando com dados ruidosos. Como é um método baseado em densidade local, alguns parâmetros devem ser definidos, como o número mínimo de pontos que constituem um cluster (MinPts) e um raio ϵ. Um ponto que possuir o número mínimo de pontos ao seu redor dentro da faixa compreendida pelo raio ϵ é considerado um ponto central, e esse conjunto de pontos, um *cluster*.

Com os pontos agrupados em *cluster*s, é esperado que esses representem a forma dos objetos procurados. Os parâmetros que fornecem a posição desses objetos no espaço são as coordenadas de seus respectivos centróides. O valor é dado por $\left\{ \bar{c} \in \mathbb{R}^3 \,|\, x = \bar{x}, y = \bar{y}, z = \bar{z} \right\}$, onde $\bar{x}, \bar{y}, \bar{z}$ são, respectivamente, as médias das coordenadas x, y e z dos pontos que constituem o *cluster*.

As técnicas previamente abordadas podem ser aplicadas com o auxílio da biblioteca *Point Cloud Library*, que possui algoritmos estado-da-arte para tratamento de dados 3D. De acordo com Aldoma et al. (2012), a biblioteca é totalmente de código aberto e está se tornando uma referência em processamento tridimensional. Outro fato que a torna interessante para o trabalho proposto é o fato dela ser completamente compatível com o ROS.

4 Metodologia

A metodologia de desenvolvimento do trabalho visa padronizar a integração da célula robótica descrita na Seção 2 por meio do desenvolvimento de uma plataforma baseada no *framework* ROS.

O processo da célula e os equipamentos responsáveis pelo controle de suas etapas foram tomados como referência para seleção dos equipamentos. Para a realização de um teste de conceito, foram escolhidos equipamentos industriais de menor escala e também equipamentos didáticos, que são comandados pelo mesmo modelo dos controladores presentes na área operacional em questão, sendo que estes serão apresentados nas próximas subseções.

4.1 Manipulador Industrial e efetuador

O manipulador industrial utilizado neste trabalho, fabricado pela ABB Corporation, conta com um braço manipulador IRB 120 e um controlador IRC5 Compact (Figura 2(a)). Em uma versão compacta, esse apresenta todas as características do controlador IRC5, que é utilizado em robôs de maior escala, fornecendo controle de movimento e alta precisão para aplicações industriais (Robotics, 2017).

(a) (b)

Figura 2: Sistema robótico e efetuador utilizados.

O efetuador BarrettHand BH8-282 (Figura 2(b)) é uma mão robótica programável que possui três dedos, com a destreza de segurar objetos de diferentes tamanhos (Barrett, 2013). A mão robótica conta com sensores tácteis na palma da mão e na ponta dos dedos. Esse efetuador pode se comunicar em padrão serial RS-232 e também via barramento CAN de alta velocidade.

4.2 Planta didática modular Festo MPS

As plantas didáticas Festo, da linha MPS, possuem arquitetura modular. Três plantas distintas serão utilizadas neste trabalho, que são as de distribuição, separação e classificação.

Cada planta é comandada por um controlador lógico programável Siemens S7-314C 2PN/DP, que é geralmente aplicado para comandar e controlar máquinas e processos industriais de médio porte. Esses controladores possuem interfaces de rede industrial MPI, Profibus DP e Profinet.

4.3 Robot Operating System

O ROS é uma ferramenta de código aberto criada para incentivar a implementação colaborativa de softwares para sistemas robóticos (Quigley et al., 2009).

Uma característica importante do ROS é a maneira com que os programas se comunicam entre si. Cada programa é chamado de nó e a troca de informações entre eles é feita por meio de tópicos e serviços, que transmitem um determinado tipo de mensagem. Outra característica relevante no ROS é a sua universalidade. É possível controlar robôs de diferentes fabricantes por meio dos vários *drivers* disponíveis.

4.3.1 ROS industrial, MoveIt! e OMPL

A aplicação da ferramenta nas áreas operacionais da Vale demanda o uso do ROS Industrial, que é um projeto de código aberto que estende as avançadas capacidades do ROS para aplicações industriais (Michieletto et al., 2014). O ROS Industrial torna possível a comunicação entre o ROS e alguns braços robóticos industriais fabricados pela ABB, Motoman, Fanuc e Universal Robots. Alguns pacotes vêm sendo desenvolvidos para comunicação com outros fabricantes (Moriano Martín, 2013).

A interação com o usuário acontece por meio do MoveIt!, que segundo Chitta et al. (2012), fornece uma plataforma de fácil utilização para desenvolvimento de aplicações robóticas avançadas, avaliação de projetos de novos robôs e construção de sistemas robóticos integrados para as áreas industriais, comerciais, pesquisa e desenvolvimento. O MoveIt! trabalha em conjunto com algoritmos estado da arte para manipulação móvel, como os fornecidos pela biblioteca OMPL, incorporando os últimos avanços em planejamento de trajetórias, manipulação, percepção 3D, cinemática, controle e navegação.

4.3.2 Sistema de integração proposto

Para flexibilizar o controle e planejamento de trajetória do braço robótico ABB IRB 120, esses são implementados no ambiente do ROS Industrial por meio da ferramenta MoveIt!, bem como a integração da mão robótica multi-dedos BarrettHand BH8-282, para auxílio na manipulação de objetos.

Os PLCs industriais que controlam a planta didática também são integrados, simulando as diferentes rotas adotadas nos processos de manipulação de amostras nos laboratórios físicos e também a troca de informações entre os mesmos.

Cada controlador têm dois tópicos do ROS associados a seus pontos de entrada e saída, que podem ser lidos e escritos pelos nós, se caracterizando por áreas de memória compartilhadas semelhantes ao sistema proposto por Martinov et al. (2016).

A programação original da planta didática é mantida, sendo o ROS responsável apenas por interfaceamento de comandos para sincronismo de tarefas. Entretanto, metodologias semelhantes as propostas por Martinov et al. (2017) e Nocoń and Choiński (2006) poderiam ser facilmente aplicadas, visto que todos os recursos para tal estão disponíveis no ROS.

O sistema de percepção funciona com base na nuvem de pontos geradas por um sensor Kinect, cujos dados serão tratados para identificação da posição e orientação das peças da planta didática Festo. Com o sistema em funcionamento, o objetivo é o desenvolvimento de uma tarefa de *pick and place*, onde o braço manipulador, com o auxílio da mão robótica, pegue uma peça por vez e alimente a planta didática. A arquitetura final do sistema está representada na Figura 3.

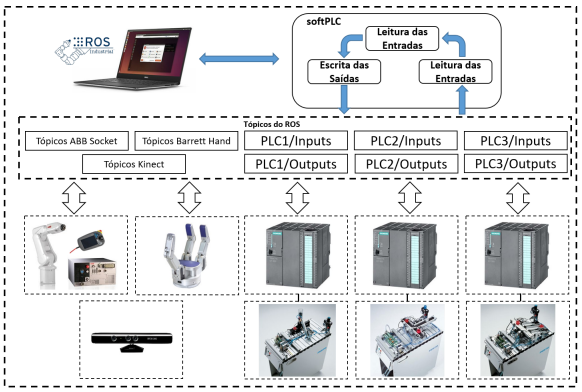


Figura 3: Arquitetura do sistema utilizando o ROS como sistema central.

5 Integração e Resultados

A seção Resultados será dividida em três subseções, onde cada uma descreverá os resultados de cada etapa proposta na metodologia, sendo esses a integração do robô e efetuador, os resultados do tratamento da nuvem de pontos para identificação da posição das peças que alimentam a planta didática, e os resultados de integração entre o ROS e os controladores industriais.

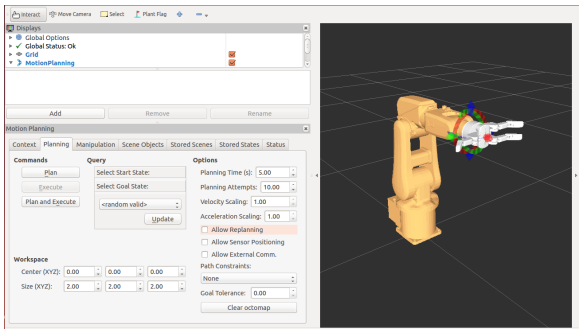
5.1 Integração entre o ROS, o Robô ABB IRB 120 e a mão robótica BarrettHand BH8-282

Esta integração utilizou o ROS como um sistema central onde um computador é responsável por processar todas as informações. A ferramenta MoveIt! disponibiliza uma interface gráfica onde diferentes trajetórias podem ser calculadas e exe-

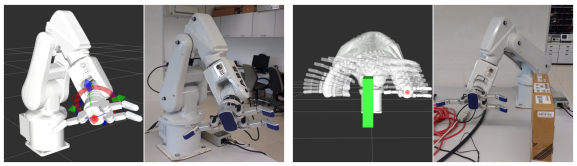
cutadas interativamente. O vetor de pontos que representa a trajetória é enviado por meio de um *socket* ao controlador IRC5 Compact, que colocará o robô em movimento seguindo a trajetória calculada. A estrutura da mensagem que trafega pelo *socket* consiste no cabeçalho da mensagem, valores e velocidades das juntas e a duração da trajetória, transmitidas em quatro *bytes* cada.

No mesmo ambiente, as juntas da mão robótica podem ser controladas diretamente. É possível, ainda, receber e tratar os valores dos sensores tácteis e de força-torque existentes no *hardware* do equipamento.

O sistema foi testado em laboratório quanto à fidelidade de execução dos movimentos, pelo sistema real, das trajetórias geradas pelo *software* MoveIt!, por meio do posicionamento interativo pela interface gráfica (Figura 4(a)) e comando para planejamento da trajetória da posição corrente até a posição objetivo (Figura 4(b)). A ferramenta de desvio de obstáculos também foi testada, incorporando um objeto no espaço entre a posição atual e objetivo. A trajetória gerada desvia do obstáculo em questão, como mostrado na Figura 4(c).



(a) Interface Gráfica do MoveIt!.



(b) Posição real e virtual (c) Desvio de obstáculo.

Figura 4: Figuras comparativas entre robô real e virtual.

5.2 Percepção 3D

As peças que alimentam a planta festo podem ser comparadas ao os copos que transportam as amostras, pois possuem geometria similar e apresentam, devido suas pequenas dimensões, um desafio maior para identificação da posição usando técnicas de percepção 3D. Estão ilustradas na Figura 5 todas as etapas do tratamento da nuvem de pontos obtida por meio do sensor Kinect, cujo o objetivo foi obter a posição de todas as peças cilíndricas para a alimentação da planta. A biblioteca *Point Cloud Library* foi utilizada para esse fim, sendo necessária a definição dos parâmetros para execução dos algoritmos. Todos os parâmetros dos filtros foram obtidos empiricamente, por meio da execução passo a passo de cada algoritmo, e os valores obtidos estão representados na Tabela 1. O pseudocódigo representando os passos para o tratamento da nuvem de pontos está descrito no Algoritmo 1.

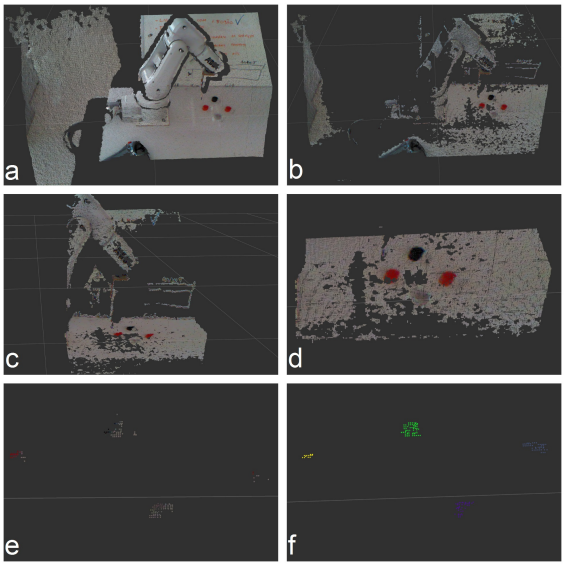


Figura 5: Etapas do tratamento da nuvem de pontos.

A Figura 5(a) mostra a nuvem de pontos bruta fornecida pelo Kinect. Na sequência, foi aplicado o filtro *Voxel Grid Downsample* (Figura 5(b)). As Figuras 5(c) e 5(d) mostram a limitação tridimensional do mapa de profundidade por meio da aplicação de filtros *PassThrough*. Após a limitação física do mapa de profundidade, foi necessário eliminar os pontos que representam a mesa onde as peças estão apoiadas.

Dessa forma, o algoritmo RANSAC foi aplicado com a função de identificação de planos. Estão ilustradas, na Figura 5(e), as peças de interesse já isoladas do mapa de profundidade original. As peças já clusterizadas podem ser vistas na Figura 5(f), onde o algoritmo DBSCAN foi aplicado para identificar regiões mais densas do mapa e atribuir uma cor randômica ao conjunto de pontos que, naturalmente, representam os objetos procurados.

Com os objetos clusterizados, o próximo passo é identificar a posição e orientação dos mesmos. A orientação é fixa para o caso estudado, pois os objetos são cilíndricos e sempre estarão dispostos sobre a mesa. Logo, foi definido que o vetor orientação é normal ao plano onde os objetos estão posicionados. A posição no espaço tridimensional é calculada com base nas médias individuais das coordenadas x, y e z dos pontos que constituem cada *cluster*.

Com a posição e orientação definidos, a fer-

Técnica	Parâmetro		Valor
Voxel Grid Downsample	leaf_size		0.026
PassThrough Filter	Eixo X	axis_min	0.0
		axis_max	0.7
	Eixo Y	axis_min	0.0
		axis_max	0.4
	Eixo Z	axis_min	1.15
		axis_max	1.5
RANSAC Segmentation	model_type		pcl.SACMODEL_PLANE
	method_type		pcl.SAC_RANSAC
	max_distance		0.0143
DBSCAN	tolerance		0.06
	min_points		10
	max_points		5000

Tabela 1: Parâmetros para processamento 3D.

ramenta MoveIt! utiliza o algoritmo *Rapidly-Exploring Random Tree* para calcular a trajetória até o objeto, onde o braço robótico realiza três movimentos para realizar o *grasp*: uma primeira aproximação, para orientar a ferramenta de modo a chegar na peça sem alterar sua posição; uma segunda aproximação, a fim de deixar a peça ao alcance da mão robótica; finalmente, a mão robótica fecha seus dedos, prendendo a peça de interesse. A Figura 6 ilustra, com mais detalhes, os passos previamente mencionados.

Algoritmo 1: Pseudocódigo para obter a posição das peças.

1: **procedure** $get\_position(cloud)$
2: $cloud \leftarrow cloud.make\_voxel\_grid\_filter(leaf\_size)$
3: $cloud \leftarrow cloud.passthrough\_filter(axis\_min, axis\_max)$
4: $outliers \leftarrow cloud.ransac\_seg(model\_type, method\_type, max\_distance)$
5: $wc \leftarrow XYZRGB\_to\_XYZ(cloud)$
6: $clusters[] \leftarrow wc.euclidean\_clustering(tolerance, MinPts, MaxPts)$
7: $centroids[] \leftarrow np.mean(clusters, axis = 0)[:3])$ **return** *centroids*
8: **end procedure**

5.3 Interfaceamento entre o ROS e PLCs Industriais

As atividades desta seção foram focadas na definição e desenvolvimento de um método para leitura e escrita de registros de memória dos controladores Siemens S7-314C - 2PN/DP, por meio do ROS. O controlador Siemens possui, como uma de suas funcionalidades, a comunicação em Modbus TCP pela sua interface Profinet.

Como este protocolo é utilizado em uma grande variedade de equipamentos de automação, o desenvolvimento permitirá estabelecer comunicação não apenas com o controlador utilizado neste teste, mas também com diversos dispositivos de outros fabricantes que utilizem o protocolo.

Para permitir que o ROS troque informações por meio do protocolo mencionado, foi utilizado um *wrapper*[1] capaz de transmitir e receber dados por meio do protocolo Modbus TCP. Basicamente, esse pacote funciona como uma ponte entre os dispositivos cliente (ROS) e servidor (Siemens S7-300). O controlador siemens foi configurado[2] para funcionar como servidor Modbus TCP, se comunicando por meio de sua porta Profinet integrada. Ao utilizar o *wrapper*, é possível obter os dados de um dispositivo que comunica via Modbus TCP e tratá-los como uma mensagem padrão do ROS. Da mesma forma, é possível publicar uma mensagem padrão do ROS para escrever valores nos registros de memória destes dispositivos.

Para expandir as possibilidades de aplicação da plataforma, foram realizados testes de comunicação com controladores ControlLogix e Compactlogix, fabricados pela empresa Rockwell, por meio de uma biblioteca[3] que fornece funções de leitura e escrita de dados destes equipamentos a partir do sistema operacional Linux e que possui total compatibilidade com o ROS. Apesar de tanto a planta didática quanto a planta real estudadas não possuírem estes equipamentos, outras células robóticas da Vale utilizam estes modelos de controladores e a comunicação com estes dispositivos é importante num horizonte de disponibilização da plataforma para uso geral. Por meio do protocolo EthernetIP, o ROS é capaz de escrever e ler áreas de memória destes controladores de forma direta.

No final dos experimentos, as tarefas de *pick and place* foram executadas com sucesso (Figura 8). Como o ponto de entrada das peças na planta didática é fixo, e todas as peças identificadas eram depositadas na mesma posição. A geometria da mão robótica favorece bastante para o sucesso do *grasp*, visto que pequenos desvios da posição são compensados pelo movimento dos dedos, que seguram a peça firmemente.

6 Conclusão

Este trabalho apresentou um desenvolvimento que visa auxiliar na resolução de problemas típicos, que podem estar presentes em diversas células com a ocorrência de equipamentos de características heterogêneas que constituam ilhas de automação, impossibilitando troca de informações e sincronismo de tarefas. Outro benefício do trabalho é a possibilidade de uma padronização para a integração de células robóticas industriais presentes na Vale. A plataforma também fornecerá uma maior flexibilidade para essas células, pois a aplicação de Percepção 3D é capaz de detectar alterações de layout e correções no posicionamento do braço robótico de acordo com variações no processo.

Os resultados preliminares mostraram que o

[1] Link para o *wrapper*: http://wiki.ros.org/modbus
[2] Guia de configuração Modbus TCP de controladores Siemens S7-300: https://goo.gl/MQbrzi
[3] Link para biblioteca e documentação *libplctag*: https://github.com/kyle-github/libplctag

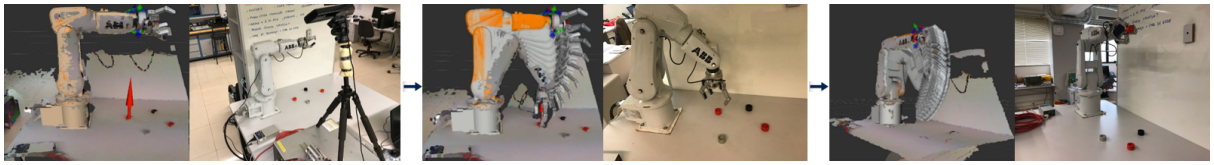
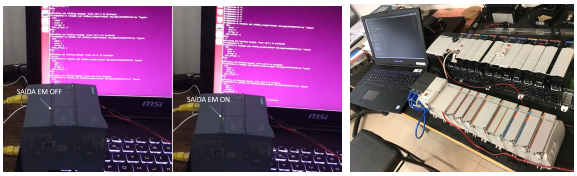


Figura 6: Etapas para *grasp* das peças de alimentação da planta didática.



(a) (b)

Figura 7: Testes de comunicação entre o ROS e controladores industriais.

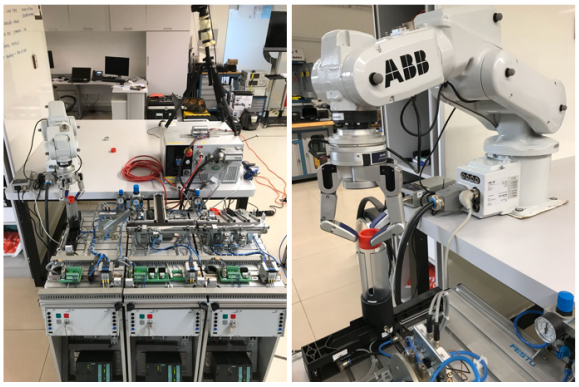


Figura 8: Tarefa de *pick and place*.

método proposto é viável, onde foi possível controlar equipamentos industriais por meio do *framework* ROS, a posição das peças da planta didática foram detectadas aplicando técnicas de percepção 3D, trajetórias foram geradas por intermédio da ferramenta MoveIt! e executadas com sucesso pelo manipulador IRB 120, bem como a sincronização das tarefas por meio da comunicação com os PLCs industriais. Esse resultado é um avanço importante, visto que todos os manipuladores industriais da ABB, independente da escala, são programados pela mesma linguagem e comandados por controladores de características semelhantes, mostrando que o desenvolvimento poderá ser facilmente replicado para células de proporções maiores.

O próximo passo dessa pesquisa consiste no estudo da robustez do sistema proposto, pois tecnologias voltadas para a industria exigem o cumprimento de padrões e normas de segurança, bem como garantias de alta disponibilidade.

Referências

Aldoma, A., Marton, Z.-C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R. B., Gedikli, S. and Vincze, M. (2012). Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation, *IEEE Robotics & Automation Magazine* **19**(3): 80–91.

Barrett (2013). *Barrett Hand BH8-282*, Barrett Technology Inc.

Bartolomeu, P., Lopes, L. S., Lau, N., Pinho, A. and Almeida, L. (2005). Integração de informação na equipa de futebol robótico cambada, *Electrónica e Telecomunicações* **4**(4): 467–477.

Chitta, S., Sucan, I. and Cousins, S. (2012). Moveit![ros topics], *IEEE Robotics & Automation Magazine* **19**(1): 18–19.

Derpanis, K. G. (2010). Overview of the ransac algorithm, *Image Rochester NY* **4**(1): 2–3.

Martinov, G., Kozak, N. and Nezhmetdinov, R. (2017). Implementation of control for peripheral machine equipment based on the external soft plc integrated with cnc, *Industrial Engineering, Applications and Manufacturing (ICIEAM), 2017 International Conference on*, IEEE, pp. 1–4.

Martinov, G., Nezhmetdinov, R. and Kuliev, A. (2016). Approach to implementing hardware-independent automatic control systems of lathes and lathe-milling cnc machines, *Russian Aeronautics (Iz VUZ)* **59**(2): 293–296.

Michieletto, S., Tosello, E., Romanelli, F., Ferrara, V. and Menegatti, E. (2014). Ros-i interface for comau robots, *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Springer, pp. 243–254.

Moriano Martín, J. (2013). Trajectory planning for the irb120 robotic arm using ros.

Nocoń, W. and Choiński, D. (2006). Fault-tolerant soft plc control using shadow processes, *IFAC Proceedings Volumes* **39**(21): 220–225.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A. Y. (2009). Ros: an open-source robot operating system, *ICRA workshop on open source software*, Vol. 3, Kobe, Japan, p. 5.

Robotics, A. (2017). *IRB 120 industrial robot*, ABB Corporation.

Rusu, R. B. and Cousins, S. (2011). 3d is here: Point cloud library (pcl), *Robotics and automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 1–4.

WIL (2018). *PCL documentation and tutorials*.

Zhou, A., Zhou, S., Cao, J., Fan, Y. and Hu, Y. (2000). Approaches for scaling dbscan algorithm to large spatial databases, *Journal of computer science and technology* **15**(6): 509–526.

Zuehlke, D. (2010). Smartfactory—towards a factory-of-things, *Annual Reviews in Control* **34**(1): 129–138.