# Robôs bípedes - Design e controle utilizando trajetórias paramétricas e arquitetura amórfica

**Alysson R. da Silva, Alexei M. C. Machado**

[1]Detartamento de Ciência da Computação – PUC Minas
CEP 30535-901 – Belo Horizonte – MG – Brazil

alysson.ribeiro.silva@gmail.com, alexei@pucminas.br

***Abstract.*** *In this work we propose an integrated approach for the design of humanoid robots, whose solution involves a simple control strategy that results on lower costs for large-scale production. A novel concept of organizational structure called Amorphic Architecture is presented. It allows for the deployment of hybrid controller systems and enables the generation of trajectories using a parameterization for the movement of the parts. The solution set also incorporates all the pipeline to generate the trajectories and their kinematics. Two parallel sub-systems for the graphical visualization of three-dimensional data are created. A small particle swarm optimization system is adapted to operate in real time, using a technique called Architecture O(1), in addition to the pipeline used to generate the paths. The experiments and hypothesis validation are performed through the construction of a compact humanoid robot and a research platform, at low costs.*

***Resumo.*** *Este trabalho apresenta uma abordagem integrada para o projeto de robôs bípedes, cuja solução envolve a simplificação do processo de controle e redução de custos para produção em larga escala. É proposta uma nova organização estrutural para a implantação de um sistema controlador híbrido, denominada Arquitetura Amórfica, que possibilita a geração de trajetórias utilizando uma parametrização para o movimento das partes. A solução criada também incorpora todo o pipeline para geração das trajetórias, contendo adaptações de algoritmos para a solução da cinemática. São criados dois sub-sistemas paralelos para a visualização dos dados de forma gráfica tridimensional e através de texto. Um pequeno sistema de otimização por enxame de partículas é adaptado para funcionamento em tempo real, utilizando-se uma técnica aqui denominada Arquitetura O(1), como complemento ao pipeline para geração das trajetórias. A execução dos testes e validação das hipóteses é feita através da construção um robô humanoide compacto e de uma plataforma de pesquisa auxiliar de baixo custo.*

## 1. Introdução

A robótica móvel é um campo de estudos relativamente novo e eminentemente multidisciplinar, cujo objetivo é o estudo, projeto, construção e manuseio de robôs que são capazes de interagir com o ambiente. Dentre os tipos de robôs existentes encontram-se os robôs bípedes humanóides — objetos mecânicos que segundo [Vukobratovic et al. 2005] caracterizam-se como antropomórficos. A construção de robôs humanoides apresenta,

como principais desafios, a geração de padrões estáveis para execução de um padrão de movimento das partes (*GAIT*), software de controle, gasto de energia e custos de produção.

Neste trabalho é apresentada uma abordagem integrada para o projeto de robôs bípedes. A solução envolve a simplificação do processo de controle e redução de custos para produção em larga escala. É proposta uma nova organização estrutural para a implantação de um sistema controlador híbrido, denominada arquitetura amórfica. Essa abordagem possibilita a geração de trajetórias, de forma simplificada, utilizando uma parametrização das fases do *GAIT*. A solução criada também incorpora todo o *pipeline* para geração das trajetórias contendo adaptações de algoritmos para a solução da cinemática. Foram criados dois sub-sistemas paralelos para a visualização dos dados de forma gráfica tridimensional e através de texto. Um pequeno sistema de otimização por *Particle Swarm Optimization* (PSO) foi utilizado e adaptado para funcionamento em tempo real, utilizando uma técnica aqui denominada *Arquitetura O(1)* como complemento ao *pipeline* para geração das trajetórias. Para a execução dos testes e validação das hipóteses foram construídos um robô humanoide compacto e uma Plataforma de pesquisa para robótica móvel (PPRM) auxiliar com custos em torno de $R\$1.200,00$.

## 2. Trabalhos relacionados

O grande desafio do controle de robôs humanoides é possibilitar que ele se desloque pelo ambiente, utilizando duas pernas sem cair e em velocidade moderada, de forma dinamicamente balanceada e consumindo a menor quantidade de energia possível [Luksch 2010]. A solução para tal desafio envolve a criação de um sistema controlador que pode ser feita através de duas abordagens: a abordagem técnica e a bio-inspirada [Luksch 2010]. A abordagem técnica utiliza teoria de controle e robótica industrial. Longe de ser perfeita, seu modelo é complexo e não responde bem a distúrbios externos desconhecidos pelo sistema. Por esse motivo, o uso das teorias do bipedalismo e estudo da morfologia humana, permitindo adaptar o modelo de mapeamento criado por nossos cérebros, possibilita a criação de robôs humanoides bio-inspirados.

O fluxo de execução dos controladores abordados na literatura envolve a utilização de três camadas: *hardware*, software e *física*, divididas em nível funcional e de decisão [Abdellatif et al. 2012]. Uma breve análise do trabalho proposto por [Brunete et al. 2012] revela meios híbridos para controle de robôs multi-configuráveis. Em [Abdellatif et al. 2012] é apresentada a utilização de *frameworks* para o auxílio no tratamento de falhas com relação ao nível funcional. Através da análise de 20 trabalhos relevantes publicados desde 1986, que tratam da arquitetura do controlador, é possível classificar um modelo de controle como sendo reativo, deliberativo ou híbrido. Grande parte dos trabalhos analisados prefere a utilização de arquiteturas híbridas, algumas mais flexíveis que outras. Propostas mais ousadas como a de [Proetzsch et al. 2010] tentam explorar com maior intensidade a utilização de sistemas bio-inspirados e a comunicação entre módulos individuais. Já em [Kherici and Ali 2014], a arquitetura através de módulos individuais é abolida, e uma abordagem por computação evolucionária usada para a execução do controle de forma integrada.

Através da análise de outros 32 trabalhos relacionados, foi observada a existência implícita de um *pipeline* para a geração das trajetórias, que auxilia o processo de

controle. Esse *pipeline* é composto pelos estágios de definição do padrão ou *GAIT* a ser utilizado, critério de estabilização, interpolação, mapeamento e execução. Em um *GAIT* completo gerado pelo *pipeline* existem duas fases simples, denominadas Fase de suporte unico (FSU), quando há apenas um pé em contato com o solo, e Fase de suporte duplo (FSD), quando os dois pés estão em contato com o solo. Essas duas fases podem ser subdivididas em sub-fases de acordo com as necessidades da solução empregada. Como exemplo, temos as soluções apontadas por [Luksch 2010] e [Furuta et al. 2001], que diferem em relação à quantidade de sub-fases. Já em [Hobon et al. 2014], essas subfases são completamente parametrizadas, constituindo-se uma abordagem mais concisa e clara. A interpretação entre a quantidade de movimentos possíveis e sua estruturação em uma cadeia manipulável varia de acordo com a abordagem para a implementação e solução dos problemas de balanceamento.

Critérios para estabilização como o *Zero Momment Point* (ZMP), *Foot Rotation Indicator Point* (FRI) e *Center of Pressure Point* (CoP), além da projeção do *Center of Mass* (CoM) para com o fecho convexo formado pelos contatos do *Support Polygon* (SuP) com o chão, tornam possível um certo grau de estabilização do robô [Vukobratovic and Borovac 2004]. Esses critérios são implementados nas soluções apresentadas por [Furuta et al. 2001] em um robô compacto, e por [Kwon and Park 2012] através do uso de uma nova estrutura morfológica de pés mecânicos. Diversos trabalhos relacionados utilizam técnicas para interpolação polinomial dos pontos chaves da trajetória constituinte de um determinado *GAIT*, baseada na definição de critérios para estabilização do robô. Dentre essas destacamos os modelos de *Bézier* e as *B-Splines* de terceira ordem. Isso possibilita a geração de trajetórias completas tridimensionais delegadas entre os planos de movimento sagital, coronal ou transversal [Hobon et al. 2014].

Neste trabalho, são discutidos diversos aspectos práticos relacionados à implantação de um controlador, a partir da proposta de um modelo simples e flexível denominado de sistema amórfico. A teoria proposta é validada através da construção completa de um robô bípede real, funcional e de baixo custo, que o difere de projetos dispendiosos como o utilizado por [Furuta et al. 2001] e os implementados por empresas de ponta como *Google*, *Boston Dynamics* e *Honda*. Na próxima seção são definidos o conceito de sistema amórfico e os procedimentos utilizados na construção do robô.

## 3. Métodos

Dentre as arquiteturas de sistemas encontradas nos principais trabalhos relacionados, destaca-se uma baseada em *Integrated Behavior-Based Control* (IB2C) [Proetzsch et al. 2010], de caráter comportamental inspirada em redes neurais, e também outra arquitetura que utiliza pequenos *codels* como código executável durante uma transição de máquina de estados [Abdellatif et al. 2012]. A análise desses trabalhos permite observar que os problemas existentes nas arquiteturas atuais estão relacionados à manipulação de funções, referências cruzadas que atuam em um sistema distribuído, falta de padronização e métricas de avaliação. Autores tendem a solucionar esses problemas por meio de técnicas de encapsulamento e conceitos de paralelismo de processos. Observa-se a duplicação de esforços para organizar um sistema distribuído baseado em eventos assíncronos, o que torna todo o processo de criação do controlador caro e redundante.

### 3.1. Sistema amórfico

Para contornar os problemas relacionados aos sistemas controladores, definimos o conceito de sistema amórfico. Um sistema amórfico é definido por um conjunto de várias sub-arquiteturas. Uma sub-arquitetura é por sua vez constituída de módulos, comportamentos, rotinas ou funções, estruturadas de forma coerente, que possibilitem o tratamento adequado para resolução dos objetivos que devem ser delegados entre os atuadores. As sub-arquiteturas podem se encontrar em dois estados, sendo estes: ativo e dormindo. Os sistemas amórficos podem possuir qualquer forma, reconfigurando-se para funcionamento em determinado contexto, ativando e desativando suas sub-arquiteturas. Isto, para certo nível de portabilidade, implica no uso de linguagens de propósito geral, deixando o mesmo passível de avaliação perante as métricas de software. Sistemas amórficos devem funcionar apenas em nível de decisão, deixando responsabilidades relacionadas ao nível funcional para algum sistema operacional presente. Um exemplo para está arquitetura pode ser visto na Fig. 1.

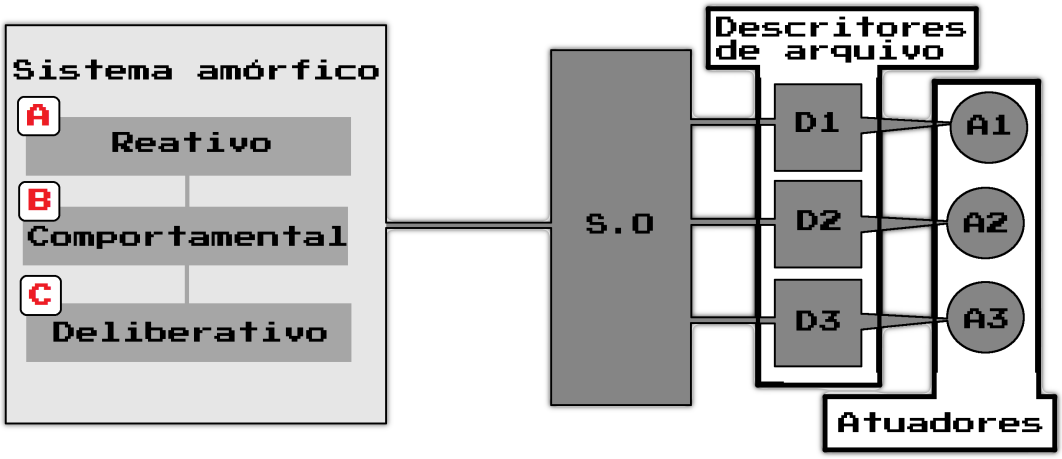


**Figure 1. Exemplo de sistema amórfico híbrido hipotético possuindo 3 sub-arquiteturas (A, B e C), que comunicam com sua camada funcional para acesso aos atuadores por meio de descritores de arquivos.**

Para auxiliar uma integração entre diversas sub-arquiteturas foram criados os seguintes critérios: (a) Comportamentos devem pertencer a apenas uma sub-arquitetura, (b) A comunicação entre sub-arquiteturas deve ser feita puramente por interfaces de comunicação, (c) O gerenciamento de condições de corrida deve ser feito pelo sistema operacional. Tal arquitetura força o acesso aos atuadores e sensores por intermédio do sistema operacional, seus descritores de arquivos e processos, o que simplifica a tarefa de gerenciamento de um sistema distribuído existente dentro do software controlador. Neste trabalho foi criado um sistema amórfico híbrido entre arquiteturas comportamentais, sistemas reativos e deliberativos. Os comportamentos utilizados possuem a mesma essência do comportamento apresentado por [Proetzsch et al. 2010]. Aspectos reativos foram possibilitados pelo meio de comunicação apresentado na seção 3.4.

### 3.2. Descrição do Robô

O robô desenvolvido neste trabalho é constituído de 16 graus de liberdade e sua estrutura física feita de alumínio. Possui $40cm$ de altura, e pesa em torno de $1.5kg$. Foram utilizados 16 servo-motores que funcionam a $6.6V$ e possuem um torque máximo de $11kg.cm$.

Todas as suas articulações são formadas por juntas de revolução, e não possui graus de liberdade com relação à cintura. São utilizados um controlador de servos (QSC32) e Arduíno controlador de periféricos (ACP). Foram instalados apenas um acelerômetro e um giroscópio acima da faixa dos ombros no plano transversal, para testes relacionados à heurística de estabilização. O robô possui elásticos adaptados para melhor estabilização postural e instalados segundo representação de grupos musculares encontradas em [Luksch 2010], os quais efetuam importante papel como estruturas auxiliares de sustentação. Sem a presença dessas estruturas, folgas entre as engrenagens ocasionam um efeito alavanca, causando menor precisão dos seus movimentos. Foram instaladas, na sola de seus pés, pequenas borrachas para amortecimento e minimização das forças de reação ao se executar um *GAIT*, de forma a evitar danos significativos na sua estrutura. Toda a comunicação é feita em tempo real por *Universal Serial Buss* (USB) com uma Central de Processamento de Dados (CPD), uma vez que limitações orçamentárias impossibilitaram a compra de um computador embarcado. A configuração de CPD utilizada resume-se a um processador Intel Core I7 4700QM, 8GB DDR3 1600MHZ, SSD 500GB, Nvidia Geforce GTX860m com sistema operacional Windows $8.1$ de $64$ bits. Para auxílio e como estrutura de apoio foi criada uma PPRM. Essa PPRM é constituída de uma esteira de $2m$ de comprimento, $1.5m$ de altura e $1m$ de largura, barras laterais para instalação de redutores de massa, necessárias devido a não utilização de fonte energética capaz de proporcionar torque máximo aos servo motores, e um motor DC sincronizado a CPD para controle da esteira. Toda sua estrutura foi construída em madeira. A fonte de energia utilizada para alimentação de todo o projeto resume-se a uma fonte ATX de $500W$, proporcionando uma alimentação de $40A$ a $5.5V$. A instalação dos redutores de massa não influencia os métodos utilizados para o controle do robô. Em toda a construção, incluindo materiais extras com exclusão da CPD, foram gastos em torno de $R\$1.200,00$, custos bastante inferiores aos apresentados por [Furuta et al. 2001] para robôs de pesquisa compactos, também de $40cm$ de altura, que estão na faixa de $R\$18.000,00$.

### 3.3. Controlador

O controlador criado neste trabalho é constituído de duas camadas: Camada de software local (CSL) e Camada de software embarcado (CSE). Tal estrutura foi utilizada devido ao uso de uma CPD não embarcada. A CSL tem como função o tratamento e gestão do controlador, do *pipeline* e para a geração das trajetórias por meio da CPD. A responsabilidade única da CSE é a obtenção de dados através do sistema sensorial, criado utilizando o ACP e seu envio para a CSL. Toda CSL é gerenciada por um Software Controlador de robô móvel de tempo real (SCRMTR) criado a partir da análise de trabalhos como o de [Furuta et al. 2001]. Seu controle de tempo real se faz necessário para propiciar um melhor ambiente que garanta comportamento antropomórfico, através de controle algorítmico. Isso caracteriza a abordagem como híbrida, ou seja, tanto bio-inspirada quanto técnica, solução que se difere de outras abordagens apresentadas por [Abdellatif et al. 2012] e [Kherici and Ali 2014] cujo foco é um controle estático. Camadas presentes no controlador e seus comportamentos podem ser vistos na Fig. 2.

O SCRMTR aqui criado é composto por dois subsistemas principais auxiliares para visualização dos dados: Subsistema de visualização tridimensional (SVT) e Subsistema gerenciador de processos para saída de dados (SGPSD). Esses sub-sistemas possuem a finalidade de proporcionar a visualização dos dados tratados durante o pipeline para ger-

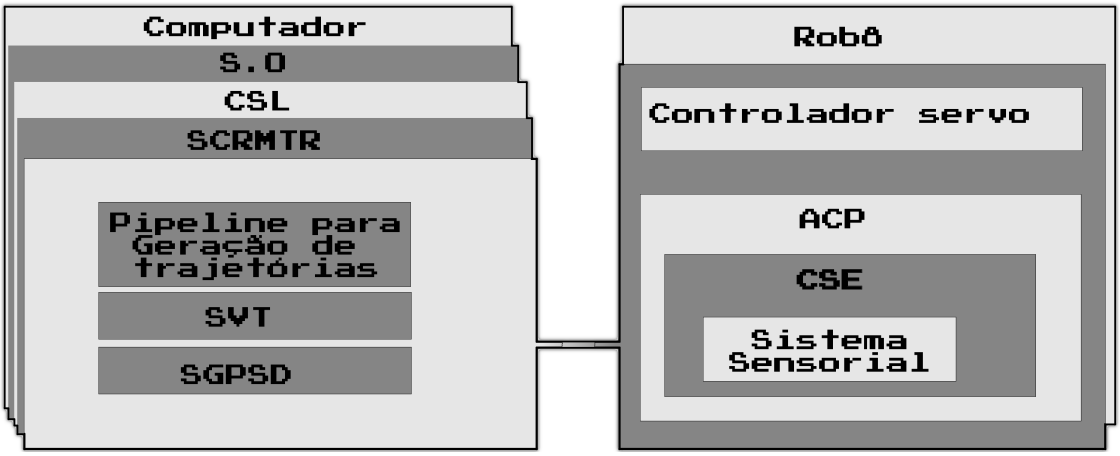


**Figure 2. Camadas de interação entre o sistema operacional, CSL, CSE e o robô.**

ação de trajetórias, o que possibilita melhor transparência do estado global do sistema em funcionamento. Como principal diferença existente entre os dois subsistemas, SVT e SGPSD, temos que o SVT foi criado utilizando-se *OpenGL* e SGPSD através de pequenas *threads* diretamente pelo sistema operacional. Todo o processo, módulo ou função pertencente ao pipeline para geração de trajetórias, criados para este trabalho, utilizam a metodologia comportamental híbrida de tempo real e satisfazem os princípios de sistema amórfico introduzido na seção 3.1.

### 3.4. Comunicação

O fluxo de dados entre o controlador e o robô físico pode ser feito por transmissão a rádio ou por meio físico através de um *remote-brain* ou *onboard-brain* [Furuta et al. 2001]. Neste trabalho foi adotada a comunicação síncrona por USB, entre CSL e CSE por um *remote-brain*. A mesma é feita através do envio de pacotes, solução padrão para comunicação em redes de computadores. O fluxo das informações de CSE para CSL é realizado através de um software embarcado que se encontra diretamente dentro da memória do ACP. Já o fluxo das informações de CSL para CSE é realizado de forma análoga, porém o formato do pacote utilizado obedece às especificações do QSC32. Para garantir certo nível de *Qualidade de Serviço* (QoS), foram utilizados pequenos *buffers* de pacotes dentro do CSL. Nenhum padrão foi utilizado com relação à criação desse pequeno aparato que possibilitou QoS para o SCRMTR. Protocolos como o próprio *Transmission Control Protocol* (TCP) não foram utilizados por proporcionarem certo *overhead* não desejável em um sistema local. Toda a tradução dos pacotes foi feita utilizando um pequeno parser descendente preditivo.

### 3.5. Controle e geração de trajetórias

O *GAIT* é identificado neste trabalho através de uma trajetória parametrizada entre o intervalo padronizado $\eta = [0 \quad 1] \quad | \quad \eta \in \mathbb{R}$, sendo uma variável $\sigma \in \eta$ criada para identificação da posição no intervalo para cada pé, abordagem semelhante à utilizada por [Hobon et al. 2014]. Um ciclo completo do *GAIT* é identificado de forma simplificada obedecendo à relação: FSD se $(\sigma_d = 1 \wedge \sigma_e = 0) \vee (\sigma_d = 0 \wedge \sigma_e = 1)$, FSU caso contrário, onde $\sigma_d$ representa o parâmetro para perna direita, e $\sigma_e$ o parâmetro para perna esquerda. Sub-fases existentes na composição do *GAIT*, como apontado por [Luksch 2010], são

identificadas por um sub-conjunto $E \subset \eta$. A geração de uma trajetória, curva tridimensional em um espaço $\mathbb{R}^3$, é feita através dos movimentos sincronizados entre os planos sagital e frontal. Foram criadas duas abordagens para sincronismo de movimentos: (a) abordagem de espelhamento para o plano sagital, onde os movimentos do lado direito são copiados e invertidos para o lado esquerdo no plano; e (b) abordagem de cópia para plano frontal, onde os movimentos de uma perna são simplesmente copiados para a outra no plano frontal. A trajetória para a cintura é gerada através da inversão da trajetória dos pés em relação ao chão no plano sagital, técnica utilizada com finalidade de simplificação. Dessa forma, não foram adotados meios para geração de trajetórias específicas para partes intermediárias como joelhos e quadril. São utilizados pontos-chave para garantir o critério de estabilização relacionado à projeção do CoM diretamente em cima do SuP para um *GAIT* estaticamente balanceado. Isso se deve a limitações da estrutura física do robô e indisponibilidade de sensores para auxílio à obtenção de critérios mais robustos como o ZMP e CoP. A obtenção e aproximação da localização do CoM é dada por uma técnica apresentada por [Lucero 2012].

Para finalização e obtenção de uma curva completa, são utilizadas técnicas de interpolação para curvas polinomiais, soluções padrão na literatura para obtenção de curvas. Através da análise de técnicas presentes no estado da arte, preferiu-se utilizar a interpolação por *B-Spline*. Pequenas alterações com o que diz respeito a seu algoritmo recursivo de interpolação, o Cox De Boor (CDB), tiveram de ser realizadas para a correção de pequenos problemas estruturais do próprio algoritmo relacionadas às suas condições de parada. Como métodos auxiliares para obtenção das poses finais para seu envio aos atuadores, foram criadas e adaptadas técnicas para solução de cinemática inversa, que segundo análises apresentadas por [Aristidou and Lasenby 2011] são as mais eficientes e que produzem poses mais realistas. Esses métodos são respectivamente o da *Jacobiana* transposta e a heurística *Forward And Backward Reaching Inverse Kinematics* (FABRIK). As adaptações se resumiram a permitir o funcionamento das técnicas em um software de tempo real, gerando-se então o Algoritmo de jacobiana transposta de tempo real (AJTTR) e o *Forward And Backward Reaching Inverse Kinematics Real Time* (FABRIKRT). O modo de locomoção, assim como os parâmetros de ajustes com o que diz respeito a qualquer configuração são controlados via sistema amórfico ou ajustados através do SVT por um observador externo. O SCRMTR juntamente com o robô desenvolvido neste trabalho para possibilitar a validação da teoria proposta, e PPRM podem ser vistos na Fig. 3. Um vídeo demonstrativo do controle e da execução de um *GAIT* pelo robô pode ser visto em https://drive.google.com/file/d/0B1ucQzq3pYAKLTg4cVM3WVFBNnM/view?usp=sharing.

### 3.6. Otimização e controle de balanço

Neste trabalho foi criado e adaptado um pequeno processo, para utilização de PSO dentro do SCRMTR para ajuste de trajetória nos planos sagital e frontal. São utilizados como função-objetivo os dados brutos recebidos do acelerômetro e giroscópio utilizados no projeto. Tal técnica possibilita a execução do PSO em um sistema de tempo real, o que em essência geraria uma técnica denominada *Real Time Particle Swarm Optimization* (RT-PSO). O único problema relacionado a isso é que normalmente essa técnica não é implementada em um sistema de tempo real verdadeiro, e sim em um *hardware* específico juntamente com técnicas que reduzem seu tempo de processamento para execução

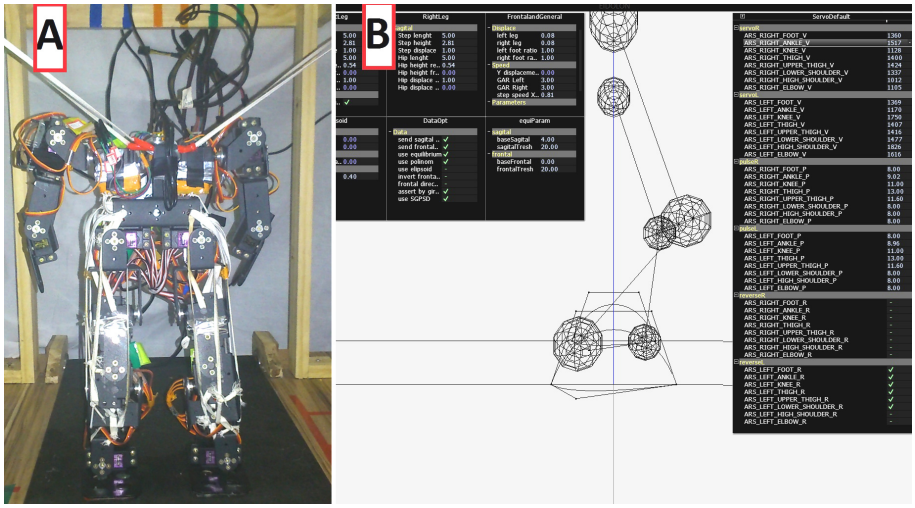


**Figure 3. A) Robô físico juntamente à PPRM e estruturas auxiliares. B) SCRMTR junto ao SVT e SGPSD em funcionamento.**

em poucos ciclos de *clock*. Em meio a sua estrutura, o PSO possui uma população, responsável por possibilitar a busca das soluções que são criadas em tempo real durante a execução de um *GAIT*. Devido a essa natureza, tal técnica aqui denominada de *Arquitetura O(1)* foi utilizada para adaptação de um *Self Regulating Particle Swarm Optimization* (SR-PSO) [Tanweer et al. 2015], satisfazendo às nossas necessidades. A técnica se baseia no conceito de que é terminantemente proibida a existência de *loops* internos no controlador. Sendo assim, todos os algoritmos e outras rotinas pertencentes a um módulo criado a partir dessa técnica possuem complexidade O(1), permitindo a incorporação de aspectos reativos no processo interno de um otimizador ou rede de comportamentos.

## 4. Resultados

Todo o sistema funcional para CPD utilizado possibilitou a execução do *loop* principal em $2511\ \mu s$, sendo considerado como módulo mais dispendioso as interfaces de comunicação, que possuem tempo de execução de $61\ \mu s$. Não foram observados *speedups* com relação à utilização do SGPSD, nem atrasos ou buracos na comunicação entre SCRMTR e ACP, tendo sido enviados $1024$ bytes de dados a cada iteração. Foi observado que o fluxo de execução entre os comportamentos varia de acordo com as referências cruzadas existentes no sistema, mesmo essas sendo *mono-thread*. Todas as estruturas extras utilizadas presentes na PPRM possuíram importante papel para realização dos testes. Os redutores de massa, apesar de influenciarem na distribuição de peso do robô, permitiram ainda assim a visualização de distúrbios causados por trajetórias instáveis. O CoM para o robô localizou-se aproximadamente a $2cm$ acima dos joelhos. O mesmo tende a ficar estacionário por consequência das abordagens de espelhamento e cópia. Todo o ajuste para possibilitar certa precisão foi feito utilizando-se funções de transformação de espaços vetoriais durante a fase de mapeamento dos domínios existentes. Houve uma grande tendência de desestabilização ao se executar a FSU, embora a redução da velocidade de caminhar e do arco da trajetória possibilitou maior estabilidade. Não houve diferença perceptível em termos de naturalidade e qualidade das poses obtidas pelos algoritmos utilizados para solução de cinemática inversa, o AJTTR tende a solucionar o problema em menos de $100$ iterações, já o FABRIKRT obtém uma solução em cerca de $3$

iterações. Com relação ao meio de andar parametrizado, era esperado que a utilização da técnica de otimização conseguisse reduzir as perturbações causadas na parte superior do robô, o que de fato ocorreu. Observaram-se quedas significativas da função objetivo entre cerca de $20$ gerações, cujo valor máximo inicial de *fitness* ficou em torno de $57$ e o mínimo observado após execução de $100$ gerações ficou em torno de $44$ para uma população de $6$ partículas. A utilização de uma técnica de otimização para este projeto não teve o intuito de obter uma solução ótima para o problema, e sim de demonstrar a possibilidade de utilização da arquitetura O(1) para otimização gradativa em tempo real em um sistema híbrido de baixo custo.

## 5. Conclusão e trabalhos futuros

Neste trabalho apresentamos uma nova abordagem para controle de um robô humanoide por meio de um sistema completamente híbrido denominado arquitetura amórfica. Para validar as hipóteses propostas foi construído um robô bípede de extremo baixo custo junto a uma plataforma de pesquisa, onde os gastos foram em torno de $R\$1.200,00$. Foi criado e implantado um sistema de tempo real, utilizando o conceito de sistema amórfico, que permite a integração de várias técnicas para construção de um sistema completamente híbrido. Simplificações das fases do meio de andar foram feitas através de parametrização. Todas as técnicas para geração de trajetórias utilizadas foram adaptadas para funcionamento em tempo real, gerando-se dois algoritmos básicos: AJTTR e FABRIKRT. Foram criados e integrados dois sub-sistemas para visualização de dados tridimensionais e textuais, assim como o conceito de Arquitetura O(1) na criação de técnica *O(1)*-SR-PSO para otimização de balanço e trajetórias nos planos coronal e sagital. Conceitos básicos para a comunicação em redes locais utilizando pacotes não padronizados possibilitaram a integração entre as diversas tecnologias de controle. A validação das hipóteses foi feita através da execução de um *GAIT* estaticamente balanceado com auxílio das estruturas existentes na PPRM.

Como trabalhos futuros estão programados o aperfeiçoamento das técnicas de mapeamento entre os domínios e a correção de erros para melhor precisão e exatidão. A melhoria no sistema de estabilização por computação evolucionária, assim como a incorporação de novos sensores, também são objetos de investigação futura. A utilização de uma abordagem completamente técnica não é desejável, porém técnicas mais robustas como ZMP poderão ser incorporadas com o objetivo de se investigarem possíveis simplificações. A simplificação do corpo físico também poderá ser aplicada em pesquisas posteriores, uma vez que é observado grande impacto da massa atual do robô na execução de seu *GAIT*.

A criação do robô apresentado neste trabalho, juntamente com os métodos de controle, não resolve todos os problemas relacionados ao controlador e à criação de um robô, sendo observados os seguintes pontos para discussão: (a) A não padronização do meio de implantação das sub-arquiteturas faz com que o processo de implementação se torne dependente do uso intenso de padrões de projeto e habilidades do codificador; (b) O mapeamento não exato presente para a musculatura artificial resulta em controle ligeiramente instável; e (c) A fonte de energia utilizada dificulta a visualização do real comportamento do sistema. Em resumo, grande parte dos problemas encontrados no projeto ocorreu por questões orçamentárias, o que não impediu a criação e execução de um controlador simplificado.

Todo o sistema foi implementado em um computador *desktop* em linguagem de propósito geral, possibilitando portabilidade. Foi registrado o funcionamento do sistema híbrido em um sistema operacional comercial, mostrando a não necessidade de criação de controladores auxiliares. Além disso, a arquitetura amórfica possibilitou a completa hibridização de um sistema complexo, propiciando o controle eficaz do robô. A validação da proposta aqui apresentada abre portas para a criação de plataformas de pesquisa de baixo custo, possibilitando a implantação de sistemas existentes no estado da arte.

## References


[Abdellatif et al. 2012] Abdellatif, T., Bensalem, S., Combaz, J., de Silva, L., and Ingrand, F. (2012). Rigorous design of robot software: A formal component-based approach. *JRAS*, 60(12):1563 – 1578.

[Aristidou and Lasenby 2011] Aristidou, A. and Lasenby, J. (2011). Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243 – 260.

[Brunete et al. 2012] Brunete, A., Hernando, M., Gambao, E., and Torres, J. (2012). A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *JRAS*, 60(12):1607 – 1624.

[Furuta et al. 2001] Furuta, T., Tawara, T., Okumura, Y., Shimizu, M., and Tomiyama, K. (2001). Design and construction of a series of compact humanoid robots and development of biped walk control strategies. *JRAS*, 37(2 - 3):81 – 100.

[Hobon et al. 2014] Hobon, M., Elyaaqoubi, N. L., and Abba, G. (2014). Quasi optimal sagittal gait of a biped robot with a new structure of knee joint. *JRAS*, 62(4):436 – 445.

[Kherici and Ali 2014] Kherici, N. and Ali, Y. M. B. (2014). Using {PSO} for a walk of a biped robot. *IJCS*, 5(5):743 – 749.

[Kwon and Park 2012] Kwon, S. and Park, J. (2012). Kinesiology-based robot foot design for human-like walking. *International Journal of Advanced Robotic Systems*, 9:259.

[Lucero 2012] Lucero, D. M. A. (2012). *Kinematic and Dynamic Analysis for Biped Robots Design*. PhD thesis, Universidad Carlos III de Madrid.

[Luksch 2010] Luksch, T. (2010). Human-like control of dynamically walking bipedal robots. Master's thesis, Universidade tecnica de Kaiserslautern.

[Proetzsch et al. 2010] Proetzsch, M., Luksch, T., and Berns, K. (2010). Development of complex robotic systems using the behavior-based control architecture ib2c. *JRAS*, 58(1):46 – 67.

[Tanweer et al. 2015] Tanweer, M., Suresh, S., and Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. *Information Sciences*, 294(0):182 – 202. Innovative Applications of Artificial Neural Networks in Engineering.

[Vukobratovic and Borovac 2004] Vukobratovic, M. and Borovac, B. (2004). Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 01(01):157–173.

[Vukobratovic et al. 2005] Vukobratovic, M., Borovac, B., and Babkovic, K. (2005). Contribution to the study of anthropomorphism of humanoid robots. *International Journal of Humanoid Robotics*, 02(03):361–387.