CENTRO UNIVERSITÁRIO FEI

LUKERCIO DE ABREU LOPES

RECONHECIMENTO DE OBJETOS BASEADO EM UMA ARQUITETURA NEURO-COGNITIVA

São Bernardo do Campo

2020

LUKERCIO DE ABREU LOPES

RECONHECIMENTO DE OBJETOS BASEADO EM UMA ARQUITETURA NEURO-COGNITIVA

Proposta de Mestrado apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Mestre em Engenharia Elétrica. Orientado pelo Prof. Paulo Sérgio Rodrigues.

São Bernardo do Campo

2020

Lopes, Lukercio.
Reconhecimento de Objetos Baseado em uma Arquitetura Neuro-Cognitiva / Lukercio Lopes. São Bernardo do Campo, 2020.
111 p. : il.

Dissertação - Centro Universitário FEI.
Orientador: Prof. Dr. Paulo Sérgio Rodrigues.

1. Arquitetura Cognitiva. 2. Reconhecimento de Objetos. 3. LIDA. 4. Inteligência Artificial Cognitiva. 5. Modelo de Mente. I. Rodrigues, Paulo Sérgio, orient. II. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

centro universitário FEI

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA EXAMINADORA

Programa de Pós-Graduação Stricto Sensu em Engenharia Elétrica

Mestrado

PGE-10

Aluno: Lukercio de Abreu Lopes **Matrícula:** 117115-6

Título do Trabalho: Reconhecimento de objetos baseado em uma arquitetura neuro-cognitiva.

Área de Concentração: Processamento de Sinais e Imagens

Orientador: Prof. Dr. Paulo Sérgio Silva Rodrigues

Data da realização da defesa: 21/02/2020

ORIGINAL ASSINADA

Avaliação da Banca Examinadora:

São Bernardo do Campo, / / .

MEMBROS DA BANCA EXAMINADORA	
Prof. Dr. Paulo Sérgio Silva Rodrigues	Ass.: _____
Prof. Dr. Ricardo de Carvalho Destro	Ass.: _____
Profª Drª Esther Luna Colombini	Ass.: _____

A Banca Julgadora acima-assinada atribuiu ao aluno o seguinte resultado:

APROVADO ☒ REPROVADO ☐

VERSÃO FINAL DA DISSERTAÇÃO

APROVO A VERSÃO FINAL DA DISSERTAÇÃO EM QUE FORAM INCLUÍDAS AS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

AGRADECIMENTOS

Gostaria de agradecer à minha mãe pelo apoio durante toda minha vida que me permitiu chegar até aqui, com todo seu esforço e suas orações. Ao meu pai, por todas as vezes que me emprestou seu carro para que eu pudesse chegar a tempo de uma aula ou reunião. À minha irmã, que sempre escutou minhas loucuras e conversas até altas horas da madrugada desde os tempos que ainda dividíamos quarto.

Não posso deixar de citar também uma grande mulher de 155,5 cm, que me apoiou, me aguentou, que sempre esteve ao meu lado a qualquer hora que precisasse, para poder me ver chegar até aqui. Obrigado Gabrielle Marques Araujo, este trabalho só foi possível pelo seu companheirismo e apoio emocional.

Agradeço ainda ao meu grande orientador, Dr. Paulo Sérgio Silva Rodrigues, ou PS, um amigo, um segundo pai, que nunca deixou de pegar no meu pé, que me inspirou durante 8 anos da minha vida a ser alguém que pudesse alcançar todos os meus maiores sonhos (sempre depois da meia-noite, principalmente aos domingos). Obrigado pela parceria, mesmo que aos 49 minutos do segundo tempo, "ou achou que eu iria desistir? - Me respeite!"

Não posso deixar de citar os amigos e colegas de trabalho, ao apoio da Capes e da FEI, que tiveram papel fundamental para o desenvolvimento deste trabalho. Obrigado a todos, espero que este trabalho seja de grande utilidade.

“The snake which cannot cast its skin has to die. As well the minds which are prevented from changing their opinions; they cease to be mind.”
Friedrich Nietzsche

RESUMO

Nas últimas três décadas, o desenvolvimento de modelos que objetivam desenvolver máquinas inteligentes tem ganho mais atenção dos pesquisadores, avançando rápidamente. Na área da Inteligência Artificial é possível identificar duas abordagens distintas. Na primeira, pesquisadores de diferentes áreas dedicam-se a desenvolver modelos cognitivos baseados em mente humana, formalizando descobertas em áreas como psicologia e neurociência, como o caso da arquitetura *LIDA*, que é talvez o exemplo mais conhecido desse tipo de iniciativa. Na segunda, buscam puramente resolver problemas a partir de modelos autônomos, não preocupando-se com toda a complexidade da mente humana, como no caso do *YOLO*, um avançado algoritmo para reconhecimento de objetos. Neste trabalho, é proposta uma abordagem utilizando-se arquiteturas cognitivas apoiadas na estrutura de modelos cognitivos já estabelecidos, em busca de uma melhora em seus resultados ao adicionar uma dinâmica que simula a mente humana. Na proposta deste trabalho, o YOLO compõe o módulo de Memória Perceptiva Associativa visando realizar o reconhecimento de objetos, tendo agregado um módulo de Memória Episódica Transiente. Este último será responsável por adicionar uma memória recente à estrutura proposta, que combinada com codelets de atenção, permite que o rastreamento de objetos ajude na decisão de quando a memória perceptiva associativa deve ser acionada, utilizando diferentes algoritmos para tais tarefas. Experimentos foram realizados sobre a base de imagens *TV77*, base esta que aglomera videos de diferentes *Datasets* conhecidas na academia. Assim, o desempenho na tarefa de reconhecer objetos foi medido e comparado com a sua implementação original, obtendo um desempenho mais rápido no processamento, sem perder de forma significativa a assertividade original do modelo.

Palavras-chave: Arquitetura Cognitiva. Reconhecimento de Objetos. LIDA. Inteligência Artificial Cognitiva. Modelo de Mente.

ABSTRACT

In the last three decades, the development of models that aim to create intelligent machines has gained more attention from researchers, rapidly evolving. In the Artificial Intelligence field two different approaches were advanced by researches. In the first approach, scientists from different areas are dedicated to develop cognitive models based on the human mind, establishing discoveries in fields related to psychology and neuroscience, such as the LIDA architecture, which is perhaps the best-known example of this type of initiative. In the second path, the researchers solve problems in a simple way from autonomous mathematical models, without worrying about using the complexity of human mind, as is possible to see with YOLO, which is an advanced algorithm for object recognition. In this work, an approach is proposed using cognitive architectures based on the structure of already established cognitive models, seeking an improvement in their results by adding dynamics that simulate the human mind. In the proposal of this work, YOLO composes the Perceptual Associative Memory module that performs object recognition, having added a Transient Episodic Memory module. The latter is responsible for adding a recent memory to the intended architecture, which combined with attention codelets, allows the object tracker to decide when the Associative Perceptual Memory should be activated, using different algorithms for such tasks. Experiments were conducted on the TV77 image database, that gather videos from different Datasets known in the academy. Thus, tasks performance after object recognition were measured and displayed with its original implementation, obtaining faster execution in processing, without losing the model's overall accuracy.

Keywords: Cognitive Architecture. Object Recognition. LIDA. Cognitive Artificial Intelligence. Model of Mind.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Computação cognitiva é um termo em ascensão na área da computação, onde diversos trabalhos abordam o tema sobre a matéria da Inteligência Artificial, ou IA. Conhecido como pai da Inteligência Artificial, John McCarthy fundou o primeiro seminário para discutir o desenvolvimento de máquinas inteligentes. Ele criou o termo em 1955, quando organizou a conferência de *Dartmouth*, com o objetivo de explorar maneiras de criar máquinas que poderiam resolver problemas como os seres humanos, considerando pensamentos abstratos, resolução de problemas e autonomicidade em adquirir novos conhecimentos (MILLER, 2003).

A área de inteligência artificial sempre buscou introduzir nos computadores o poder de decisão sem que houvesse um pré-mapeamento de todas as possibilidades para a realização de tarefas apresentadas a uma máquina. Em 1960, um movimento dentro do âmbito de IA se desenhava de forma natural, que se tornaria no futuro a área de estudos sobre cognição em computadores. Pesquisadores de diferentes universidades criaram grupos interdisciplinares de estudos, os nomes variaram, mas os objetivos eram os mesmos, juntar esforços da psicologia, linguística, antropologia, neurociência e ciência da computação para desenvolver estudos que dedicariam à realizar formulações computacionais para as descobertas relacionadas a mente humana, como descrito em (MILLER, 2003).

Em paralelo, as áreas desenvolveram-se. A Inteligência Artificial, em sua maior parte, se moldou na resolução dos problemas, utilizando métodos puramente matemáticos. Mesmo sob uma tentativa de modelar neurônios, revisto em (GARDNER; DORLING, 1998), e se aproximar do encéfalo humano, os estudos em geral não se atentaram a dinâmica complexa da mente humana em sua totalidade.

O desenvolvimento das Redes Neurais Artificias é um exemplo clássico desta distinção. Mesmo com avanços, o método provou-se limitado quando comparado a magnitude do encéfalo humano, mesmo se mostrando promissor na tarefa de reconhecimento de padrões. Por outro lado, tal limitação não barrou o desenvolvimento de novas soluções.

O resultado disso foi uma revolução na computação, com o desenvolvimento de soluções desde sistemas de recomendação até reconhecimento de textos em linguagem natural, de voz, e objetos em imagens e videos (LOPS; GEMMIS; SEMERARO, 2011) (CAMBRIA; WHITE, 2014) (GAIKWAD; GAWALI; YANNAWAR, 2010) (RUSSELL; NORVIG, 2016).

Métodos recentes de reconhecimento de objetos estão sendo amplamente estudados e aprimorados, sempre comparados em competições e desafios sobre bases de imagens, como em

(LIN et al., 2014) (EVERINGHAM et al., 2010) (DENG et al., 2009). Estas competições já demonstraram o alto desempenho dos algoritmos e o rápido avanço da área, porém os métodos apresentados ainda não fazem grande correspondência com modelos da mente humana.

Por sua vez, diversos trabalhos surgiram na tentativa de mapear comportamentos, regiões e a estrutura do cérebro humano na sua amplitude. Esta área, também conhecida como Inteligência Artificial Geral, se desenvolveu propondo modelos de mente humana, preocupados em modelar desde partes biológicas, até funções específicas das regiões cerebrais (SAMSONOVICH, 2010).

O desenvolvimento destes modelos segue sob rápida evolução, onde funções conhecidas da mente humana foram ou estão sendo descritas, em paralelo ao desenvolvimento e aprimoramento de seus respectivos modelos computacionais (SAMSONOVICH, 2010).

Em busca de unir novamente ambas as áreas, este trabalho propõe uma junção de um avançado modelo de detecção de objeto sob uma arquitetura cognitiva baseada nas diferentes áreas que compõe a mente humana.

Neste modelo proposto, a arquitetura *LIDA* (*Learning Intelligent Distribution Agent*, ou em português, Agente de Aprendizagem Inteligente Distribuída) é utilizada como base cognitiva, onde os estudos desenvolvidos por seus pesquisadores (revisado no Capítulo 4), serão aplicados a um modelo de reconhecimento de objetos em tempo real. Uma vez que o modelo *LIDA* preocupa-se em modelar as funcionalidades da mente humana, e não cérebros, um modelo baseado em uma rede neural convolucional pode ser utilizada para compor um de seus módulos.

Para o presente trabalho foi escolhido o algoritmo de reconhecimento de objetos *YOLO, You Only Look Once* (em português, Você Olha Apenas uma Vez), revisado na Seção 3.2.2. Ele está alinhado com o estado da arte em relação ao seu desempenho perante a outros trabalhos. Uma vez que foi inspirado na habilidade humana de rápidamente olhar e reconhecer objetos em uma cena, será utilizado como núcleo de reconhecimento para o módulo responsável pela interpretação de imagens proposto pela arquitetura *LIDA*, a Memória Perceptiva Associativa.

Ainda com base na arquitetura *LIDA*, será utilizado um módulo de Memória Episódica Trânsiente. Este, será responsável por adicionar uma memória recente à dinâmica do *YOLO*, aproveitado-se de fatos ocorridos em um passado recente, objetivando uma melhora em sua acurácia.

1.1 OBJETIVO

Aplicação no reconhecimento de objetos da arquitetura cognitiva *LIDA* (*Learning Intelligent Distribution Agent*), baseando-se no algoritmo *YOLO* (*You Only Look Once*).

Desta forma, apresentar uma nova abordagem para a área de reconhecimento de objetos, introduzindo a dinâmica de comportamento da mente humana no desenvolvimento de soluções de visão computacional.

1.2 ESTRUTURA DO TRABALHO

Este trabalho foi organizado sequencialmente, encadeando os assuntos do início ao fim. A partir do próximo capítulo são apresentadas explicações dos trabalhos relacionados, seguidos pelos conceitos fundamentais para o desenvolvimento deste trabalho. Nos capítulos subsequentes, o leitor poderá acompanhar a medologia e a proposta experimental para o projeto. A seguir, é apresentada uma breve descrição de cada capítulo.

Capítulo 2 tem como base a explicação de fundamentos utilizados para o desenvolvimento do objetivo deste trabalho. Estes fundamentos introduzem ao leitor, conceitos de áreas como modelos cognitivos e reconhecimento de objetos.

Capítulo 3 apresenta projetos de outros autores que são relacionados a esta obra. A bibliografia exposta neste capítulo é tomada como base e fundamentação para o andamento deste trabalho.

Capítulo 4 apresenta mais detalhes sobre a arquitetura cognitiva base para este trabalho, o *Lida*, entendendo seu ciclo e o funcionamento de seus módulos, tanto conceituais quanto os já formalizados.

Capítulo 5 detalha a metodologia utilizada, demonstrando como foram realizadas as implementações deste projeto, com exemplos dos módulos propostos e sua dinâmica de funcionamento.

Capítulo 6 ilustra como foi desenvolvida a proposta experimental desta pesquisa, indicando os parâmetros assumidos sobre os algoritmos, as comparações que foram realizadas e a base de dados a ser utilizada.

Capítulo 7, os resultados experimentais são apresentados, com o auxilio de gráficos e tabelas.

Capítulo 8, por fim, tem o papel de expor as conclusões finais obtidas pelo autor deste trabalho.

2 CONCEITOS FUNDAMENTAIS

Neste capítulo são descritas as definições encontradas nas áreas sobre modelos cognitivos e reconhecimento de objetos, onde os modelos e métodos que serão referenciados no decorrer deste trabalho são apresentados.

A seguir, são explorados diferentes conceitos, estudados pela ciência da computação, psicologia e neurociência. Tais conceitos, foram tomados como base para o desenvolvimento deste trabalho, por meio de seus desdobramentos sobre reconhecimento de objetos e a arquitetura *LIDA*, esta última vista em mais detalhes no Capítulo (4).

2.1 RECONHECIMENTO DE OBJETOS

O reconhecimento de objetos é uma técnica de visão computacional para identificar objetos em imagens ou vídeos, por meio de dois passos globais, a detecção dos objetos de interesse na cena e o reconhecimento dos mesmos. Por sua vez, é tido como uma das aplicações fundamentais dos algoritmos de aprendizagem profunda e aprendizado de máquina.

Esta é uma tecnologia fundamental por trás dos carros autônomos, o que lhes permite reconhecer um sinal de parada ou distinguir um pedestre de um poste de luz. Também é utilizado em uma variedade de aplicações, como inspeção industrial, visão robótica, além de identificação de doenças em imagens médicas de forma geral, incluindo análise patológica em ressonância magnética, tumografias computadorizadas, etc.

2.1.1 Delimitação de Objetos

Os métodos de reconhecimento de objetos, em geral, trabalham em duas fases. Inicialmente é realizada a localização do objeto de interesse. Nesta fase, estimativas são calculadas sobre regiões da imagem em busca de localizar objetos dominantes pertencente a um grupo de imagens previamente conhecido. Para realizar a delimitação de áreas da imagem, diferentes abordagens foram propostas, dois métodos aplamente difundidos foram os utilizados em (SERMANET et al., 2013) e (REN et al., 2015).

Em (SERMANET et al., 2013), um conjunto fixo de janelas quadradas distribuidas uniformemente sobre as imagens (como uma grade) foi proposto. Este método aproveita a operação de convolução para regredir rapidamente e classificar os objetos na forma de janelas, ou caixas,

deslizantes sobre as imagens. Por sua vez, (REN et al., 2015) infere regiões que potencialmente possuem objetos de interesse, tendo melhor acurácia no ajuste das delimitações aos objetos, porém sendo duas vezes mais lento que o primeiro método.

Na Figura (1) é possível verificar as caixas delimitadoras, mais conhecidas como *bounding boxes*, localizando objetos de interesse nas imagens. As caixas em vermelho representam o padrão-ouro (ou *ground truth*) previamente conhecido em contraste com as caixas em amarelo indicando a localização estimada realizada por um algoritmo de localização de objetos.

Por sua vez, na segunda fase ocorre o reconhecimento dos objetos delimitados pelas *bounding boxes*. Uma classificação do que há dentro de cada caixa delimitadora é realizada indicando a estimativa calculada e o nível de confiança para cada classificação. Neste momento, os algoritmos de reconhecimento de objetos deveriam classificar como pássaros os exemplos da Figura (1).

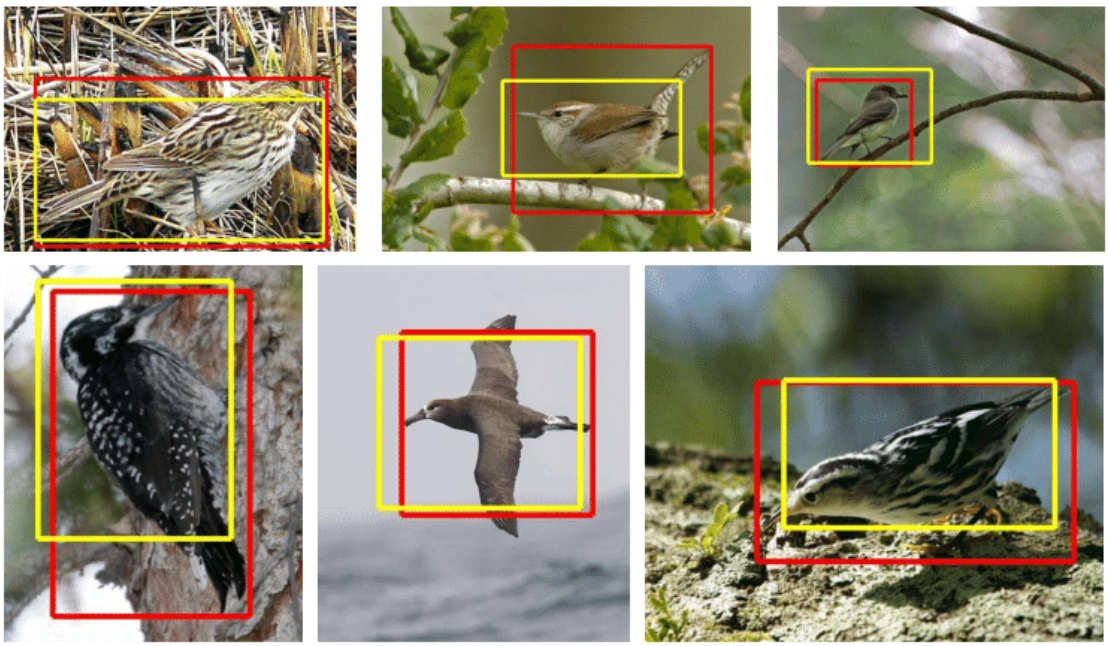


Figura 1 – Exemplo de Localização de Objetos em imagens

Para delimitar áreas específicas das imagens, o método de *bounding boxes* é amplamente utilizado no reconhecimento de objetos. São delimitadas áreas da imagem de entrada que potencialmente contém objetos de interesse. Uma *bounding box* pode ser representada como um vetor de 4 elementos, armazenando suas coordenadas de borda $(x_0, y_0, x_1, y_1) \in R$, ou armazenando sua localização central, além de sua largura e altura, representados respectivamente por $(x, y, w, h) \in R$. Uma *bounding box* é geralmente acompanhada por uma pontuação de confiança da probabilidade de a caixa conter um objeto.

2.1.2 Similaridade

Para avaliar a qualidade das *bounding boxes* inferidas, uma medida de distância é aplicada sobre os vetores calculados e o padrão-ouro (ou *ground truth*), sendo ela o índice de Jaccard, também conhecido como IoU (*Intersection over Union*).

o IoU é uma estatística utilizada para comparar a similaridade e diversidade de conjuntos de amostras, como visto em (DENŒUD; GUÉNOCHE, 2006). O coeficiente de Jaccard mede a similaridade entre conjuntos de amostras finitas e é definido como o tamanho da interseção dividido pelo tamanho da união dos conjuntos de amostra, definido na Equação (1).

$$J(A,B) = \frac{|A \cup B|}{|A \cap U|} = \frac{|A \cup B|}{|A| + |B| - |A \cap U|} \tag{1}$$

Para as *bounding boxes*, o coeficiênte é calculado utilizando a área de intersecção entre a caixa inferida e o padrão-ouro (ou *ground truth*), sobre a união de ambas as áreas, como ilustrado na Figura (2).

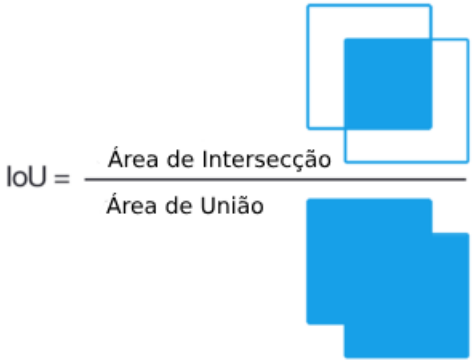


Figura 2 – Métrica IoU calculada para *Bounding Boxes*

2.1.3 Detecção de Mudança de Cena

Diferentes algoritmos podem ser utilizados para compara uma imagem com outra. Neste trabalho os seguintes algoritmos foram considerados para desempenhar este papel, por conta de sua simplicidade e velocidade de execução.

No modelo de Correlação, quanto mais próximo de 1, mais parecidos são os histogramas. Uma equivalência perfeita resultaria em um valor 1 (BRADSKI; KAEHLER, 2008). Caso o resultado seja 0, indica uma associação aleatória, ou uma não correlação. Caso o resultado seja -1, uma diferença máxima entre os histogramas ocorreu.

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}$$

$\bar{H_k} = \frac{1}{N} \sum_J H_k(J)$ onde N equivale ao número de posições em um histograma.

Para o método Chi-Quadrado, quanto mais próximo de 0, mais próximos são os histogramas comparados. Existe uma total diferença entre eles, quando o resultado tende a infinito (BRADSKI; KAEHLER, 2008).

$$d(H_1, H_2) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)}$$

Já o método de interseção indica boas combinações para resultados altos, enquanto resultados próximos a 0 indicam total falta de compatibilidade (BRADSKI; KAEHLER, 2008).

$$d(H_1, H_2) = \sum_I \min(H_1(I), H_2(I))$$

Por fim, a distância de Bhattacharyya é aproximada a partir da distância de Hellinger, utilizando um fator de normlização. Neste caso, quanto mais baixo o resultado, mais parecidos são os histogramas (BRADSKI; KAEHLER, 2008).

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H_1}\bar{H_2}N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}$$

2.1.4 Rastreamento de Objetos

O objetivo do rastreamento é prever onde um objeto previamente conhecido se encontra no *frame* atual. Como o rastreamento se dá sobre um conjunto de quadros, ao ser acompanhado é possível indicar como o objeto está se movimentado. Com isso, é possível indicar a direção e a velocidade de como um objeto está se movendo analisando uma pequena área da imagem. Isso ocorre, pois os algoritmos de rastreamento codificam a aparência do objeto, verificando uma pequena área em torno do objeto em uma sequência de *frames*. A seguir, os algoritmos utilizados neste trabalho são brevemente apresentados.

O algoritmo *Boosting* precisa ser treinado em tempo de execução com exemplos positivos e negativos do objeto (BRADSKI; KAEHLER, 2008). A *bound box* é tomada como exemplo positivo para o objeto, e regiões de imagem fora da caixa delimitadora são tratados

como exemplo negativo. Dado um novo quadro, o classificador é executado em todos os pixels nas proximidades do local anterior e a pontuação do classificador é registrada. A nova localização do objeto é aquela em que seu valor é máximo.

Já o algoritmo *MIL (Multiple Instance Learning)*, não são especificados exemplos positivos e negativos, mas coleções positivas e negativas (BRADSKI; KAEHLER, 2008). O conjunto de imagens na coleção positiva não é um exemplo positivo. Em vez disso, apenas uma imagem na coleção positiva precisa ser um exemplo positivo. Assim, mesmo que a localização atual do objeto rastreado não seja precisa, quando amostras da vizinhança da localização atual são colocadas no conjunto positivo, há uma probabilidade considerável de que essa coleção contenha pelo menos uma imagem na qual o objeto esteja centralizado.

O KFC *Kernelized Correlation Filters* baseia-se nas idéias apresentadas nos dois rastreadores anteriores. Este rastreador utiliza o fato de que as múltiplas amostras positivas usadas no rastreador MIL têm grandes regiões sobrepostas. Esses dados sobrepostos levam a algumas propriedades matemáticas que são exploradas por esse modelo para tornar o rastreamento mais rápido e preciso (BRADSKI; KAEHLER, 2008).

TLD *Tracking, learning and detection*, ou em português rastreamento, aprendizado e detecção. Como o nome sugere, esse rastreador decompõe a tarefa de rastreamento de longo prazo em três componentes - rastreamento (curto prazo), aprendizado e detecção. No artigo do autor (KALAL; MIKOLAJCZYK; MATAS, 2011), ele cita que o rastreador segue o objeto de quadro a quadro. O detector localiza todas as aparências que foram observadas até o momento e corrige o rastreador, se necessário. O aprendizado estima os erros do detector e o atualiza para evitar esses erros no futuro.

O *MEDIANFLOW* rastreia o objeto nas direções para frente e para trás no tempo e mede as discrepâncias entre essas duas trajetórias. Minimizar esse erro do *ForwardBackward* permite detectar falhas de rastreamento de maneira confiável e selecionar trajetórias confiáveis nas sequências de vídeo (BRADSKI; KAEHLER, 2008).

A soma mínima de erro quadrático da saída (MOSSE, *Minimum Output Sum of Squared Error*) usa correlação adaptativa para rastreamento de objetos, que produz filtros de correlação estáveis quando inicializados usando um único quadro. O rastreador MOSSE é robusto a variações de iluminação, escala, posição e deformações não rígidas (BRADSKI; KAEHLER, 2008). Ele também detecta a oclusão com base na razão *pico de lobos laterais*, que permite ao rastreador pausar e retomar de onde parou quando o objeto reaparece.

No algoritmo DCF-CSR, (*Filtro de Correlação Discriminativa com Confiabilidade de Canal e Espacial*), é utilizado o mapa de confiabilidade espacial para ajustar o suporte do filtro

à parte da região selecionada do *bound box* (BRADSKI; KAEHLER, 2008). Isso garante o aumento e a localização do objeto selecionado e o rastreamento aprimorado das regiões ou objetos não retangulares. Ele usa apenas 2 recursos padrão *(HoGs e Colornames)* (BRADSKI; KAEHLER, 2008).

2.2 CONSCIÊNCIA

A consciência é um assunto intrinsicamente complexo, sendo seu conceito um conjunto de várias ideias, por conotar diferentes fenômenos (GÖK; SAYAN, 2012). Não existe uma definição concreta do que seja consciência, mas ao longo dos anos diferentes definições foram apresentadas, contribuindo para estudos aprofundados que abrangeram a filosofia e são hoje aplicados em muitas áreas da ciência.

(VELMANS, 2009) diz que o termo "consciência" refere-se à experiência em si. Em vez de sermos exemplificados por algo específico que observamos ou experimentamos, ela é exemplificada por todas as coisas que observamos e experimentamos. (TONONI, 2011) complementa se referindo à consciência como "sinônimo de experiência - qualquer experiência - de formas ou sons, pensamentos ou emoções, sobre o mundo ou sobre o próprio eu". A consciência possui muitos significados, por isso, muitos estudos focam em parcelas da consciência para resolução de problemas.

A exemplo, (BAARS, 2002) por meio da teoria do espaço global (GWT), propõe que a consciência seja a porta de entrada para o cérebro. A teoria do espaço global sugere que a consciência permite que várias redes cooperem e competem na resolução de problemas, como a recuperação de itens específicos da memória imediata.

A consciência possui muitas funções. Ajuda a lidar com situações novas ou problemáticas para as quais não tem resposta automática. Permite executar tarefas que exigem conhecimento de localização, forma, tamanho ou outras características de objetos no ambiente (FRANKLIN, 2003). Neste trabalho, conceitos como *Global Workspace* e suas memórias periféricas serão introduzidos junto ao processo de reconhecimento de objetos.

2.2.1 Consciência Funcional

Consciência funcional desempenha um papel importante como um filtro perceptivo que permite ao agente se concentrar apenas nas informações mais relevantes. Sendo assim, ajuda

na seleção de ações, permitindo que o agente reúna recursos, a fim de escolher o que fazer em seguida e assim resolver problemas de maneira eficiente (FRANKLIN, 2003).

2.2.2 Consciência Fenomenal

Consciência fenomenal refere-se à sensação subjetiva da experiência consciente (SUN, 1997). O conhecimento de cor ou de um som é uma experiência consciente inexplicável. Por exemplo, não é possível explicar a uma pessoa que não tem visão desde o nascimento o que é a cor purpura, pois, a consciência fenomenal é impossível de ser transmitida por palavras, mas deve ser experimentada para ser entendida (FRANKLIN, 2003).

2.2.3 Consciência Artificial

A consciência de máquina ou consciência artificial é inspirada na filosofia, psicologia e neurociência, assim como compartilha muito dos objetivos da inteligência artificial. Sendo assim, não é um campo unificado com um conjunto de objetivos claramente definidos. Atualmente, pesquisadores de diferentes áreas trabalham em aspectos distintos do problema, dificultando muitas vezes a compreensão de como tudo se encaixa (GAMEZ, 2008).

(MANZOTTI; TAGLIASCO, 2008) afirma que a consciência artificial é proposta como uma área interdisciplinar, cujo objetivo consiste em formular teorias e modelos a serem implementados em computadores e em estruturas robóticas. Existem dois tipos de consciência artificial: forte e fraca. A consciência artificial forte origina um agente que tenha consciência genuína. Por sua vez, a fraca origina uma máquina que pode se comportar como uma consciência genuína.

Por fim, (PRASAD; STARZYK, 2010) define: “Uma máquina é consciente se, além dos mecanismos necessários para percepção, ação, aprendizagem e memória associativa, possui um executivo central que controla todos os processos (conscientes ou subconscientes) da máquina”.

Assim sendo, o desenvolvimento de conciências artificiais trazem aos computadores comportamentos mais próximos dos seres humanos, possibilitando a inserção de variáveis antes ignoradas no desenvolvimento de algoritmos de inteligência.

2.3 CIÊNCIA COGNITIVA

A ciência cognitiva (SUN; BOOKMAN, 1994) é um objeto de estudo de diversas áreas, entre elas a ciência da computação. A cognição é o processamento de informações através de sentidos para adquirir conhecimentos, e o propósito de estudo da ciência cognitiva é compreender o funcionamento da mente humana.

O principal tipo de abordagem que será vista neste trabalho é a simbólica, que consiste na cognição poder ser representada por fórmulas e modulações matemáticas, sendo assim, podem ser replicadas para modelos computacionais.

2.4 ATENÇÃO

A atenção é um conceito fortemente relacionado à consciência, especulando-se que a consciência tem um surgimento biológico a partir da atenção (GRAZIANO, 2014), sendo esse conceito frequentemente utilizado no cotidiano como concentração mental sobre algo específico. Contudo, para neurociência (GRAZIANO; WEBB, 2014; GRAZIANO; WEBB, 2015), a atenção é o processo cognitivo de seleção de informações que serão processadas através de uma competição para selecionar quais destas serão realizadas mais profundamente, sendo essa conhecida como "competição tendenciosa", na qual os sinais que envolvem essa competição podem ser influenciados por estímulos *top-down* e *bottom-up*. Uma vez que o estímulo seja atendido este vence a competição, porém, a atenção é um estado em constante mudança que as vezes ocorre de uma representação visual ganhar a competição do momento.

2.5 GLOBAL WORKSPACE THEORY

O *Global Workspace Theory* (GWT) é um modelo de uma arquitetura cognitiva de consciência, proposto por (BAARS, 1997), com o objetivo de explicar os processos conscientes e inconscientes que ocorrem no cérebro. A teoria sugere que a consciência está relacionada à “capacidade limitada do cérebro”; isto é, a memória imediata e à capacidade seletiva de atenção, visto que essa capacidade limitada provém de uma ligeira impressão onde o cérebro é um órgão lento que executa as tarefas sequencialmente. Contudo, quando ele realmente é estudado, vemos que, na verdade, o cérebro é um órgão com um conjunto complexo de redes neurais, trabalhando em paralelo. Cada rede com sua especialidade cognitiva, sendo estes processos inconscientes e trabalhando de forma eficaz. O GWT considera a consciência como o acesso

global ou o ato de transmissão entre os processos cognitivos e as memórias. Podemos comparar esse acesso global com um antigo conceito de inteligência artificial no qual ele foi baseado, o "quadro negro". Esse conceito é uma base comum a diferentes processos específicos.

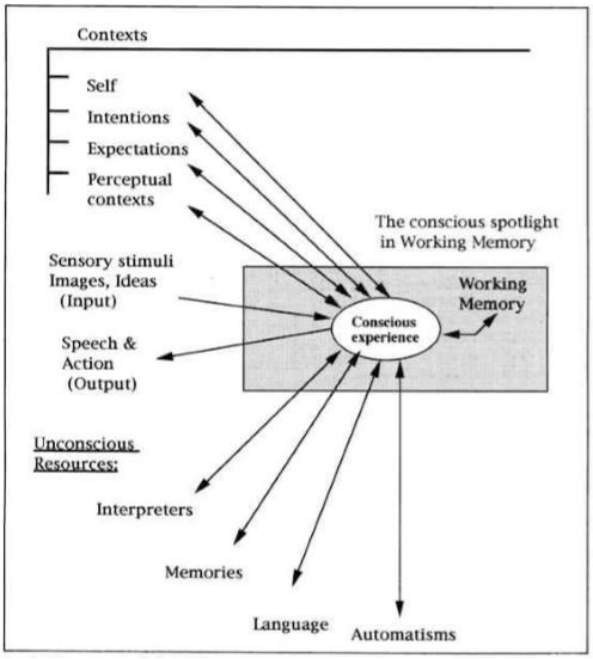


Figura 3 – Diagrama do *Global Workspace*, retirado de (BAARS, 1997; BAARS, 2005)

O funcionamento dessa teoria pode ser exemplificada pela metáfora do teatro da consciência ou da mente, apresentada na Figura (3), em que o teatro é escuro e inconsciente. A consciência é tida como pontos de luz no palco, orientados por um holofote de atenção obedecendo a uma certa sequência. O holofote de atenção é o que guia a consciência, tanto voluntariamente quanto espontaneamente, como um ponto de luz no palco. Portanto, apenas eventos que estejam nessa área luminosa são rigorosamente conscientes, existindo assim, uma competição entre os pensamentos, sensações ou imagens, com o intuito de alcançar o palco e a parte luminosa de modo que o vencedor dessa competição torne-se consciente. Nos bastidores do teatro há contextos que geralmente são inconscientes e influenciam fortemente nos processos conscientes, moldando os eventos conscientes. O diretor trabalha invisivelmente nos bastidores, sendo aquele que toma as decisões baseado em metas. Por fim, temos o público, no qual a consciência é uma porta para uma vasta área de processos cognitivos inconscientes, como memória de longo prazo, linguagem, interpretadores e automatismos.

3 TRABALHOS RELACIONADOS

Por conta da composição deste trabalho, este capítulo está dividido em duas partes. A primeira tratará de trabalhos relacionados no campo da AGI, *Artificial General Intelligence*, ou em português, Inteligência Artificial Geral, mais especificamente sobre as principais arquiteturas cognitivas da área. Em seguida, este capítulo tratará de modelos de reconhecimento de objetos em imagens, mais especificamente, dos métodos mais utilizados para processamento de imagens na literatura recente. Ambas as áreas serão exploradas ao longo deste trabalho buscando aplicar conceitos já conhecidos na área de reconhecimento de objetos com o uso de arquiteturas cognitivas, ainda pouco exploradas na computação

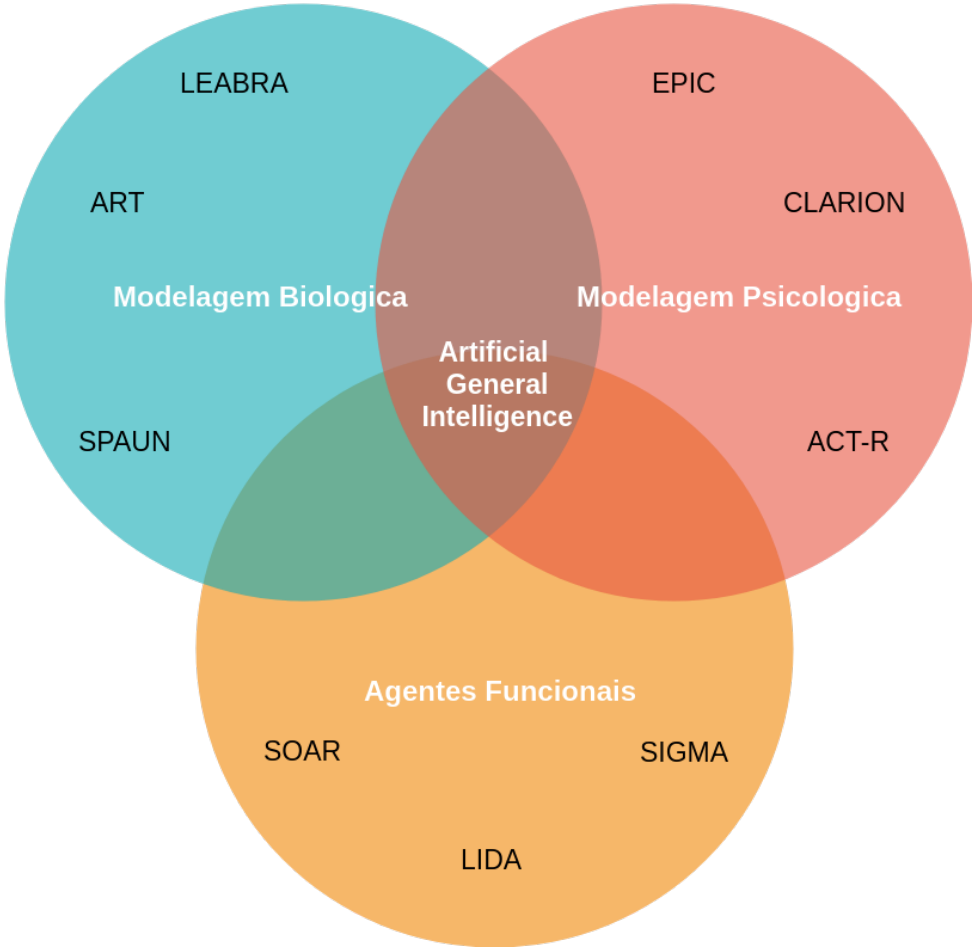
A arquitetura *LIDA (Learning Intelligent Distribution Agent)*, tem importância fundamental para a realização deste trabalho, sendo usada amplamente para as propostas aqui apresentadas, portanto será explorada a parte em mais detalhes no Capítulo (4).

3.1 ARQUITETURAS COGNITIVAS

A evolução da neurociência e da psicologia permitiram uma melhor compreensão do encéfalo humano. A partir disso, modelos foram propostos em busca de demonstrar o funcionamento da mente humana em sua complexidade, trazendo diferentes interpretações representadas em arquiteturas com aspectos que hoje já apontam semelhanças (LANGLEY; LAIRD; ROGERS, 2009).

O desenvolvimento dessas arquiteturas segue sob rápida evolução, onde funções conhecidas da mente humana foram ou estão sendo descritas, em paralelo ao desenvolvimento e aprimoramento de seus respectivos modelos computacionais (SAMSONOVICH, 2010). Tais modelos fazem parte da área de pesquisa conhecida como *Artificial General Intelligence*, ou AGI, onde alguns deles serão apresentados neste capítulo.

Figura 4 – Exemplos de Arquiteturas de Consciência ativas por foco.



Fonte: Autor

Allen Newell, um dos precusores da área, em (NEWELL, 1994) define que arquiteturas cognitivas especificam aspectos da cognição humana que permanecem constantes durante todo o tempo de vida de um agente. Ele ainda propõe que as principais hipóteses sobre cognição devem trabalhar em conjunto na resolução de problemas, resolvendo pontos específicos de forma linear durante a execução de uma tarefa.

Em busca de um modelo próximo ao que já foi descrito pela neurociência e psicologia, seja de forma empírica ou científica, arquiteturas foram propostas sob diferentes perspectivas de pesquisadores. Na Figura (4), as três principais abordagens são ilustradas (Modelagem Biológica, Modelagem Psicológica e Agentes Funcionais), classificando os respectivos exemplos de modelos cognitivos, dentre eles, *SOAR, LIDA, SIGMA, ACT-R, CLARION e ART* (LANGLEY; LAIRD; ROGERS, 2009).

No nível biológico, estão agrupados exemplos de arquiteturas que focam em modelar a baixo nível aspectos já conhecidos, que influenciam no processo de tomada de decisão. O

foco de tais arquiteturas está atrelado a aspectos como, os tipos modelos de interação entre os neurônios (em diferentes níveis), o ritmo em que ocorrem as sinapses, a quantidade de neurônios utilizada, entre outros detalhes, buscando sempre manter a fidelidade a modelos biológicos como no caso do modelo *ART* em (GROSSBERG, 1987).

Em outro grupo de arquiteturas cognitivas, já mais acima em seu nível de granularidade, encontram-se os modelos psicológicos, como é o caso do *EPIC* em (KIERAS; MEYER, 1997). Estes modelos tentam preservar, com certa fidelidade, as áreas conhecidas do encéfalo, mas focando em sua organização e como se dão a interação entre essas áreas, tentando prever erros que humanos cometem, seus comportamentos e o tempo de reação para realizar determinadas ações.

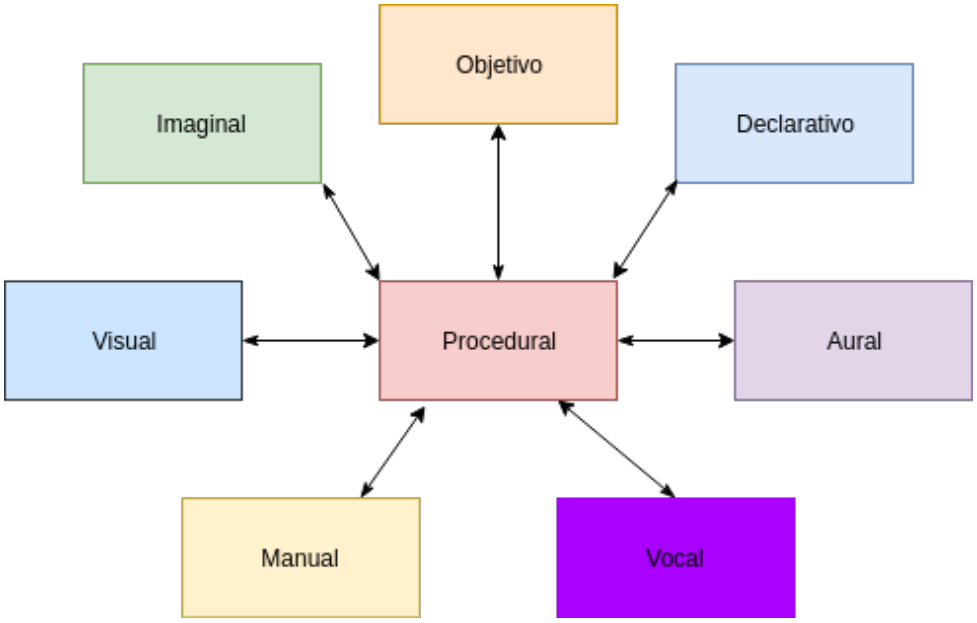
Por fim, temos o conjunto de modelos baseados em agentes funcionais. Estas arquiteturas estão interessadas principalmente em produzir sistemas funcionais, trazendo artefatos que buscam solucionar problemas mais complexos e não apenas tentando simular partes já conhecidas do encéfalo humano. No caso da arquitetura *SOAR*, suas aplicações variam desde tomada de decisão até processamento de linguagem (LAIRD; ROSENBLOOM; NEWELL, 2007).

Neste trabalho utilizaremos o modelo de consciência *LIDA* (FRANKLIN; JR, 2006), que compartilha de aspectos deste último subconjunto de arquiteturas cognitivas descrito. A seguir, são apresentados os trabalhos considerados como mais relevantes pelo seus respectivos números de citações e/ou identificados pela tabela comparativa de arquiteturas cognitivas implementadas (SAMSONOVICH, 2010), da área conhecida como *Artificial General Intelligence*. É introduzido ao menos um trabalho de cada subconjunto apresentado pela Figura (4).

3.1.1 ACT-R

Em meados da década de 70, pesquisadores na universidade de Carnegie Mellon propuseram uma arquitetura cognitiva para representar conhecimento e experiência baseada em conceitos da psicologia. Utilizando-se de módulos quase que totalmente independentes, a *ACT-R*, ou *Adaptative Control of Thought - Rational*, em português Controle Adaptativo do Pensamento Racional, o modelo provê informação para *buffers* dedicados, que possuem capacidades limitadas (Anderson, 1976). Na Figura (5), cada caixa representa um módulo responsável por um processo específico, sendo controlados por um módulo central.

Figura 5 – Visão panorâmica do ACT-R. Cada caixa representa um módulo controlado por um outro chamado procedural central, cada um com capacidades limitadas.



Fonte: Adaptado de (SAMSONOVICH, 2010)

A informação de entrada é codificada e separada em partes, seguindo as chamadas regras de produção. Tais regras processam os blocos acessando e configurando os mesmos nos *buffers* do módulo. Para este modelo, a implementação original foi realizada em *LISP*, incluindo uma interface, que é opcional, desenvolvida em *TCL/TK* (ANDERSON, 2009) . Recentemente, outras implementações utilizando diferentes linguagens e *frameworks* foram disponibilizadas pela comunidade de desenvolvedores (ANDERSON; LEBIERE, 2014).

A estrutura do *ACT-R* suporta recursos e componentes que são comuns para outras arquiteturas cognitivas, sendo eles a memória semântica que é codificada como blocos e a memória procedural. Sistemas de memória de trabalho e episódicos não são explicitamente definidos pelos autores, porém o conjunto de *buffers* utilizados na comunicação entre módulos pode ser considerado como provedor de capacidades de memória de trabalho conhecidas no encéfalo humano (SAMSONOVICH, 2010) (Anderson, 1976). Os módulos sensoriais, motores e outros específicos incluem a entrada visual proposicional, baseada em partes, entrada auditiva também proposicional, baseada em partes, além de funções motoras básicas para o controle de mãos e dedos, esta última visando aplicações na robótica (ANDERSON, 2009).

Dentre os algoritmos de aprendizado utilizados no *ACT-R* estão o aprendizado por reforço, que é utilizado para regras de produção fazendo uso de uma implementação com desconto linear e atualização Bayesiana, para recuperação de memória. A arquitetura permite formar no-

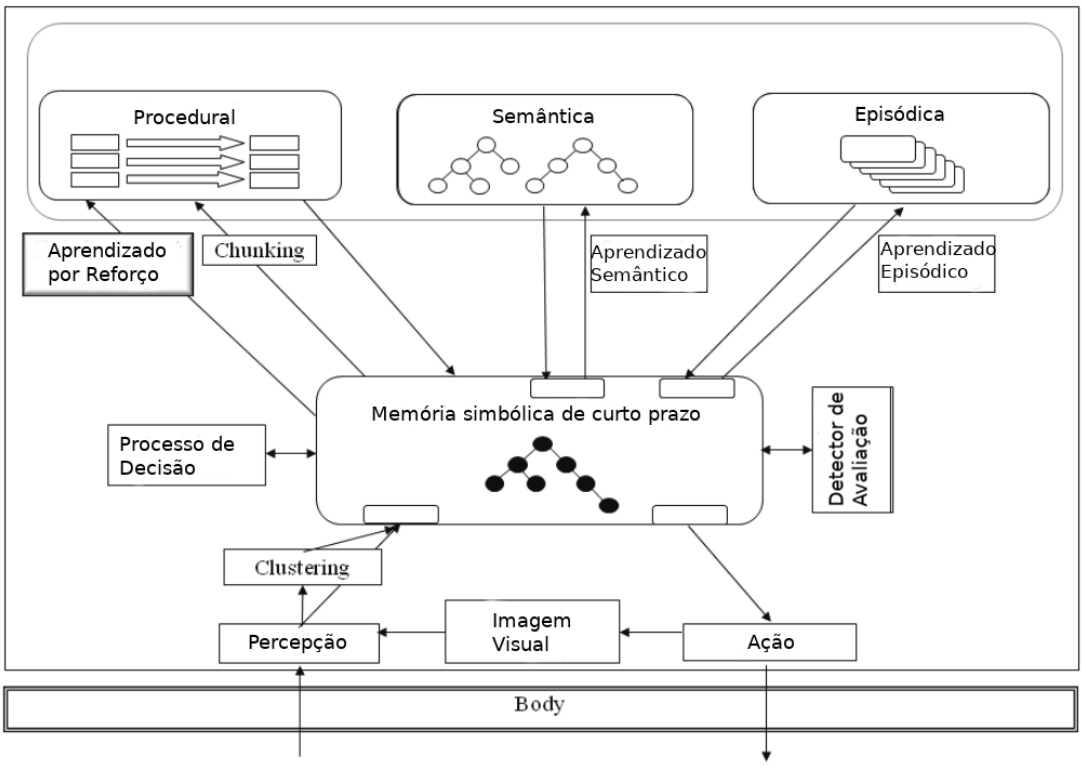
vas produções a partir das existentes, além de aprendizado automático por blocos utilizando o conteúdo encontrado no *buffer* (ANDERSON; LEBIERE, 2014).

3.1.2 SOAR

Em (LAIRD; NEWELL; ROSENBLOOM, 1987) é apresentada uma das mais importantes publicações sobre arquiteturas cognitivas, chamada *SOAR*. Inicialmente, por conta de suas características, seu nome indicava, em inglês, "*basic cycle of State, Operator, And Result*", porém abandonou o acrônomo após a evolução da arquitetura.

A *SOAR* baseia-se na chamada "Hipótese do Problema Espacial" descrita por completo posteriormente em (NEWELL, 1994). Esta hipótese sustenta que todo comportamento orientado por objetivos pode ser atingido utilizando uma busca por possíveis estados, em meio a um espaço problema. Em cada passo, um único operador é selecionado e, em seguida, aplicado ao estado atual do agente, o que pode levar a alterações internas, como a recuperação de conhecimento da memória de longo prazo e modificações ou ações no mundo externo (LAIRD; NEWELL; ROSENBLOOM, 1987).

Figura 6 – Visão panorâmica da versão extendida da arquitetura cognitiva SOAR



Fonte: Adaptado de (SAMSONOVICH, 2010)

Para representar experiências e conhecimento, a arquitetura *SOAR* utiliza-se de regras de conhecimento procedural, memória episódica e conhecimento semântico por uma estrutura de grafo relacional. Ao longo dos anos, a teoria sobre a arquitetura foi aprimorada e extendida, contendo modelos de memória de trabalho, memória semântica, imagens mentais e aprendizagem por reforço (LAIRD; ROSENBLOOM; NEWELL, 2007). Tais componentes podem ser visto na Figura 6.

A *framework*, originalmente implementada em C possui interface para quase qualquer linguagem, além de contar com uma versão em *Java* desenvolvida mais recente (SAMSONOVICH, 2010). Esta conta com componentes comuns a outras arquiteturas cognitivas como memória de trabalho (*Workspace*), memória semântica, memória episódica, memória procedural, memória icônica, sistema de recompensa e *codelets* de atenção de consciência (LAIRD; ROSENBLOOM; NEWELL, 2007).

Diferentes estudos de agentes funcionais aplicam a arquitetura *SOAR* em áreas como processamento de linguagem, tarefas de memória de trabalho e raciocínio analógico limitado. Dentre os exemplos de tarefas solucionadas pela arquitetura estão a resolução da torre de Hanoi, exploração espacial, aprendizado por instrução, aprendizado e navegação, estas duas últimas ainda não comparáveis ao comportamento humano (SAMSONOVICH, 2010).

3.1.3 ART

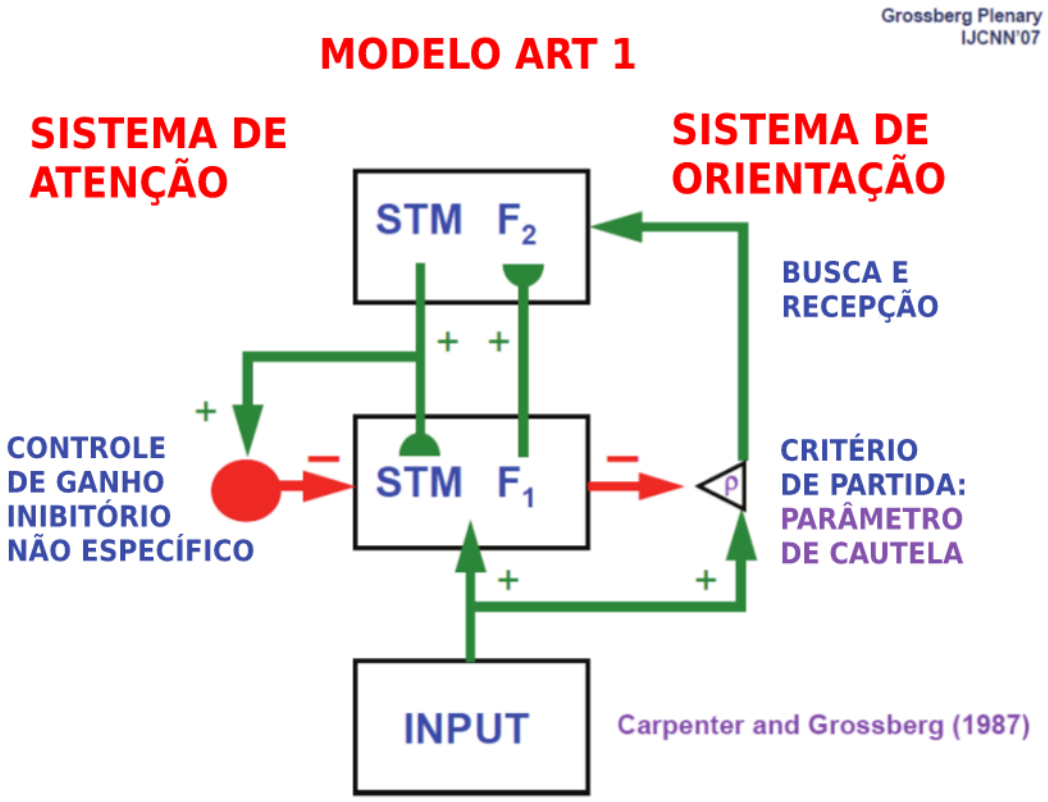
Dentre as arquiteturas cognitivas inspiradas na biologia, a *ART* (do inglês *Adaptative Resonance Theory*, ou em português Teoria de Resonância Adaptativa), foi apresentada em (GROSSBERG, 1987) e pode ser considerada uma das mais relevantes em termos de citação.

Pertencendo à classe de arquiteturas bioinspiradas, um de seus objetivos é tentar exclarecer como os indivíduos se adaptam autonomamente, em tempo real, em um mundo em constante mudança sobre efeito de eventos inexperados (SAMSONOVICH, 2010). Além disso, tentam explicar como os diferentes tipos de aprendizagem se comportam quando expostos à reorganização sofrida por tal dinamismo.

Em (GROSSBERG, 2013) o autor revisa a evolução da arquitetura, explorando seus recursos e componentes. É demonstrada uma alta granularidade de seus módulos individuais, cada qual sendo descrito por meio de um baixo número de equações. Esta abordagem implica em uma quantidade elevada de módulos descritos, como parte da busca por uma arquitetura desacoplada, capaz de controlar diferentes tipos de inteligência, refletindo assim a complexidade de um encéfalo humano. Por conta dessas características, o modelo pode ser computacional-

mente representado, mantendo inclusive, fidelidade na disposição de processamento, onde os módulos propostos são executados paralelamente desde sua concepção (GROSSBERG, 1987).

Figura 7 – Visão panorâmica da arquitetura cognitiva ART



Fonte: Adaptado de (SAMSONOVICH, 2010)

Na Figura (7), experiências e conhecimentos são representados por meio dos principais componentes da arquitetura, incluindo representações de regiões do encéfalo, circuitos do tálamo e córtex laminar, em meio a um sistema de atenção e outro de orientação (GROSSBERG, 1987). Além disso, implementações para estudos experimentais foram realizadas utilizando redes neurais não-lineares com *feedback*, escalas temporais e espaciais (SAMSONOVICH, 2010).

Para manter a fidelidade com o sistema humano, diversos recursos estão presentes seguindo modelagens biológicas conhecidas (GROSSBERG, 2013). Para a memória de trabalho, aplicam-se desvios recorrentes centralizados e fora dos arredores da rede, que obedece a invariância da *Long-Term Memory (LTM)*, ou em português, Memória de Longa Duração. A memória semântica possui associações limitadas entre seus componentes, enquanto a memória episódica é restrita, baseando-se em representações espaciais e temporais do hipocampo. O sistema de recompensa é modelado como a amígdala, o hipotálamo e os gânglios basais, interagem com o córtex sensorial e pré-frontal para aprender a direcionar atenção e ações para objetivos valoriza-

dos, sendo usado para ajudar a explicar dados sobre condicionamento clássico e instrumental, transtornos mentais (autismo, esquizofrenia) e tomada de decisão sob risco (SAMSONOVICH, 2010). A arquitetura *ART* prevê uma ligação entre processos de Consciência, Aprendizagem, Expectativa, Atenção, Ressonância e Sincronia (*CLEARS*), tendo todos os estados conscientes como sendo estados ressonantes.

3.1.4 IDA

O IDA, ou *Intelligent Distributed Agent*, é um agente autônomo apresentado e desenvolvido por Stan Franklin nos artigos (FRANKLIN; KELEMEN; MCCAULEY, 1998) e (FRANKLIN, 2003), que é capaz de lidar com situações novas de maneira mais flexíveis e inspiradas nos seres humanos. Para que isso seja possível, a implementação da consciência do agente segue as diretrizes da *Global Workspace Theory* (GTW). Entretanto, o agente simula apenas uma consciência funcional.

O IDA foi desenvolvido em conjunto com a Marinha Norte Americana, com o objetivo de ter um software que automatiza o processo de tomada de decisão para designar marinheiros para seus novos postos de trabalho. Ao se comunicar com os marinheiros por e-mail, da mesma forma que um humano faria, o IDA deveria negociar novos empregos e, eventualmente, fazer atribuições com base nas políticas da Marinha e nas preferências dos marinheiros.

Ele é divido em diferentes módulos com funções específicas. A seguir, um breve resumo sobre cada um deles, considerando sua aplicação final na Marinha:

a) Percepção: consiste no processamento de e-mails trocados com os marinheiros em linguagem natural, para que seja possível reconhecer, categorizar e entender o objetivo da mensagem, sem que exista um padrão fixo de respostas.

b) Workspace: novos conteúdos, vindos da percepção e de processos internos ao IDA (suas memórias periféricas), são enviados ao *workspace*, de modo que possam ser encaminhados a processos menores e especializados. Esse processo ocorre com o intuito de encontrar rotinas. Todo conteúdo armazenado no *workspace* dissipa-se com o tempo e pode ser sobrescrito, uma vez que possui características de memória temporária.

c) Memória Associativa: associa memórias, emoções e ações com dados vindos da percepção. Para realizar isso ela aplica, majoritariamente, a memória esparsamente distribuída como memória associativa de longo prazo.

d) Emoções: demonstra o quão bem o sistema IDA como um todo está realizando suas tarefas. Por exemplo, ele pode entender que demorou para responder uma solicitação de um marinheiro e isso pode influenciar em sua seleção de próximas ações, buscando compensar o atraso gerado.

e) Mecanismo Consciente: é o módulo responsável por decidir quais são os conteúdos que devem tornar-se conscientes. Para isso, a atenção vigia os dados que estão no *workspace*, avalia a importância de cada um e envia aqueles selecionados para serem processados.

f) Seleção de Ação: para realizar a seleção de ação o IDA faz uso das *behavior nets*. Estas possibilitam uma seleção de ação de alto nível.

g) Satisfação da Restrição: define medidas que dimensionam o grau de adequação de um trabalho a um marinheiro. Cada preferência, problemática e política são valoradas e, então, é feita uma soma ponderada para dimensionar o quanto tal função se adéqua a um marinheiro neste momento.

h) Deliberação: cria possíveis cenários e escolhe estre eles qual se encaixa melhor na atual situação.

i) Ação voluntária: um mecanismo interno ao IDA que permite ações serem executadas quando se tornam conscientes, exceto quando ela gera ideias contrárias e propostas novas.

j) Negociação: o agente envia uma lista de funções ao marinheiro e negocia com ele até que uma delas seja aceita. Para que a negociação se torne viável, o agente deve conseguir tomar decisões e responder a mensagens em linguagem natural.

k) Metacognição: regula o conhecimento do agente, possibilitando que ele saiba avaliar ações cognitivas, objetivos e estratégias.

l) Aprendizagem: o agente IDA possui uma aprendizagem declarativa. Isso ocorre graças ao fato da Memória Associativa ser uma memória esparsamente distribuída e, por isso, ela aprende associações como efeito colateral de sua estrutura.

O primeiro agente IDA foi desenvolvido para uso da marinha americana, com a finalidade de fazer uma melhor distribuição de atividades para seus marinheiros. Ele se comunica com os marinheiros em linguagem natural via e-mail, entende o conteúdo do que é enviado a ele e responde como se fosse um ser humano, assim como acessa uma base de dados e compreende o conteúdo ali armazenado. É dever dele alocar o marinheiro de acordo com suas preferências

e qualidades, o que a leva a ter que negociar com o marinheiro via e-mail, também em linguagem natural. As pessoas responsáveis por essa tarefa relataram que o agente realizava todo o processo de forma similar ao que a própria pessoa faria.

3.1.5 CLARION

O CLARION (*Connectionist Learning with Adaptative Rule Induction On-line*), em português Aprendizagem Conexionista com Indução de Regra Adaptável On-line, é um modelo cognitivo que usa técnicas de redes neurais e *machine learning*, beaseada em modelagem psicologica. A ideia de aprendizado no modelo ocorre de forma simultânea e concorrente, que caracteriza a maneira que o ser humano assimila informações, apresentado em (SUN, 1997).

A arquitetura consiste em dois níveis principais, sendo eles o superior e o inferior, respectivamente acessível e inacessível. O nível superior, contém interações com o mundo externo, focado em obter informações do momento. A parte inferior do CLARION corresponde ao comportamento, e nele está armazenado o conhecimento por meio de redes com representações distribuídas. Ambos os níveis são interconectados e cooperam entre si para a tomada de decisões e aprendizagem (SUN; FRANKLIN, 2007). A representação do modelo pode ser vista na Figura 8.

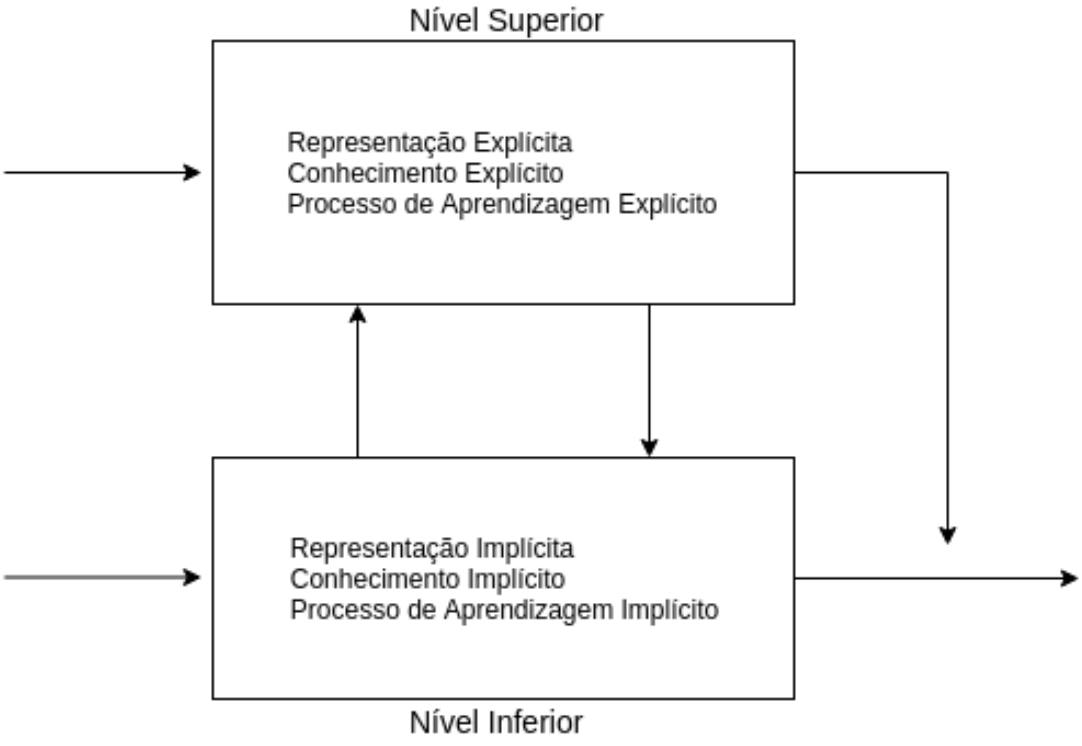


Figura 8 – Arquitetura CLARION, adaptado de (SUN, 1997)

O modelo opera da seguinte forma (SUN, 1997): um determinado estado é observado, e enquanto isso o nível inferior é associado com possíveis ações a serem tomadas por meio de atribuição de valores. As possíveis ações de nível superior são descobertas de acordo com a entrada e com as regras do ambiente. Após esse procedimento, as ações classificadas do nível superior são comparadas com as ações do nível inferior, para assim escolher determinada ação do nível inferior. Realiza a ação escolhida, e se necessário, utiliza um recurso de reforço. Depois, atualiza os valores denominados Q – obtidos pelo algoritmo de aprendizado por reforço *Q-Learning-Backpropagation*, que avalia a qualidade da ação - e as regras do ambiente, estas com o *Rule-Extraction-Refinement*, para o método adaptar as regras se não apresentarem sucesso na ação. E assim, sucessivamente, a medida que novos estados ocorrem.

3.2 RECONHECIMENTO DE OBJETOS

Quando os humanos olham para uma fotografia ou assistem a um vídeo, podem identificar pessoas, objetos, cenas e detalhes visuais. Os algoritmos de reconhecimento têm por objetivo ajustar um modelo matemático para fazer o que é natural aos seres humanos, ou seja, obter um nível de compreensão do que uma imagem contém. Este trabalho utilizará um método de reconhecimento de objetos remodelado para uma arquitetura neuro-cognitiva, portanto, esta Seção trará produções da área de reconhecimento de objetos relacionadas a este trabalho.

Recentemente, técnicas de aprendizado de máquina e aprendizado profundo tornaram-se abordagens populares para problemas de reconhecimento de objetos. Ambas as técnicas aprendem a identificar objetos em imagens, tendo diferentes aplicações em diversos setores, como: permitir que robôs domésticos localizem objetos em uma casa, identificação do mundo externo para carros autônomos, detecção de pessoas e rostos, além de auxilio na identificação de orgãos ou tumores em imagens médicas (SCHMIDHUBER, 2015).

3.2.1 Arquiteturas de Reconhecimento

A publicação de (DUDA; HART, 1973) é uma das primeiras revisões dos princípios teóricos para técnicas propostas, até então, tanto na classificação de padrões quanto na análise de cenas. Ele reúne técnicas previamente espalhadas por toda a literatura e fornece uma notação comum concisa que facilitou a compreensão e comparação dos muitos aspectos do reconhecimento de padrões ainda em seus conceitos matemáticos. O conteúdo foi dividido em cinco áreas de assuntos inter-relacionados de relevância até os dias atuais, sendo eles: Seleção de Funcionalidades, Classificação Livre de Distribuição, Classificação Estatística, Aprendizagem Não Supervisionada e Aprendizagem Sequencial.

Na década de 1980, com a evolução dos computadores, estudos práticos sobre clusterização se iniciaram com o objetivo de detectar objetos de interesse em imagens, como descrito em (STOCKMAN; KOPSTEIN; BENETT, 1982). Os autores desenvolveram uma técnica, aplicada e testada em cartografia para detectar objetos em mapas de satélites. A técnica utiliza uma variação de conjunto de pares de *features* das imagens do modelo e combina com informações locais dentro das imagens. Para cada possível par de *features* correspondentes, transformações *RST* (rotação, dimensionamento e conversão) são aplicadas. A clusterização aplicada sobre diferentes conjuntos de parâmetros *RST* mostrou uma boa generalização global do modelo. São

apresentados resultados que demonstraram, para a época, que a técnica não requer detecção sofisticada de recursos e é robusta em relação a mudanças de orientação e conteúdo da imagem.

Estudos mais avançados surgiram em meados da década de 1990. Em (JAIN; RATHA; LAKSHMANAN, 1997), os autores utilizaram filtros Gabor para segmentação e detecção de objetos. Uma imagem é passada por um filtro pra simétrico Gabor. As imagens que passaram pelo filtro são selecionadas e submetidas a uma transformação não-linear sigmoidal. Então, uma medida de energia sobre texturas é computada em uma janela ao redor de cada pixel das imagens transformadas. A energia da textura (*feauture* de Gabor) e suas localizações espaciais são inseridas em um algoritmo de agrupamento de erro quadrático. Este algoritmo de agrupamento produz uma segmentação da imagem original atribuindo a cada pixel na imagem um rótulo de agrupamento que identifica a quantidade de energia local média que o pixel possui em diferentes orientações e frequências espaciais.

O método é aplicado a várias imagens visuais e infravermelhas, cada uma contendo um ou mais objetos. A região correspondente ao objeto é geralmente segmentada corretamente, e um valor único calculado sobre as *features* de Gabor é associado à segmentação dos objetos de interesse (JAIN; RATHA; LAKSHMANAN, 1997).

Figura 9 – Exemplo de objetos detectados em (LOWE, 1999)

Em 1997, estudos ligando computação e neurociência são publicados relacionando diretamente áreas do encéfalo humano com algoritmos propostos. Em (LOWE, 1999), um sistema de reconhecimento de objetos foi desenvolvido utilizando uma nova classe de *features* locais de imagens. Tais *features* são invariantes ao dimensionamento, translação e rotação das imagens

e parcialmente invariantes a alterações de iluminação e projeção, uma representação pode ser vista na Figura (9). Os autores indicaram que essas características compartilham propriedades similares com neurônios no córtex temporal inferior que são usados para reconhecimento de objetos na visão de primatas (LOWE, 1999).

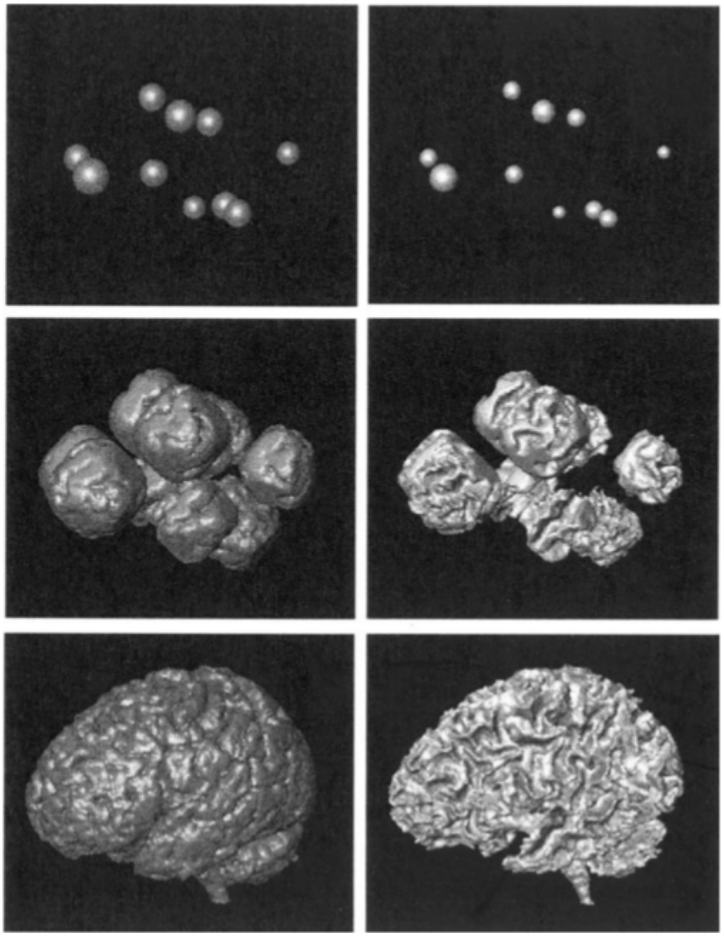


Figura 10 – Resultados do algoritmo Coupled-surfaces propagation, retirado de (ZENG et al., 1999)

As *features* são detectadas por meio de uma abordagem de filtragem em etapas que identifica pontos estáveis no espaço. São criadas chaves, indicadas pelos pequenos quadrados na Figura (9), para as imagens que permitem deformações geométricas locais sem que a detecção seja perdida em múltiplos planos de orientação e em múltiplas escalas de tamanho. As chaves são usadas como entrada para um método de indexação do vizinho mais próximo que identifica correspondências com objetos candidatos. A verificação final de cada correspondência é obtida por meio de uma solução de mínimos quadrados residuais para os parâmetros do modelo desconhecido. Resultados experimentais mostraram que o reconhecimento era robusto mesmo em imagens parcialmente ocluídas, com um tempo de processamento de até 2 segundos. Este método veio a ficar conhecida como *SIFT (Scale-invariant feature transform)* (LOWE, 1999).

Com o foco na área médica, os autores apresentam em (ZENG et al., 1999), uma diferente abordagem de propagação de superfícies acopladas. A implementação do método introduz uma inicialização de baixa complexidade, além de eficiência computacional e capacidade de capturar dobras sulcais profundas. Inicializando a partir de esferas concêntricas, obtém-se as fronteiras corticais interiores e exteriores, segmentando a massa cinzenta cortical e a substância branca. Além disso, isola-se o tecido encefálico do tecido não-encefálico. Na Figura 10, o algoritmo de propagação de superfícies acopladas é aplicado sobre a mesma imagem do cérebro, mas com um conjunto diferente de esferas de inicialização. Para os resultados finais (com inicializações diferentes), a taxa de *True* Positivo (TP) de um em relação ao outro foi superior a 99,5%, e a taxa de Falso Positivo (FP) foi inferior a 0,5%, o que demonstra a robustez do algoritmo para a base de testes utilizada.

Em (DALAL; TRIGGS, 2005) os autores deram um grande passo na área de detecção de objetos utilizando algoritmos de aprendizagem de máquina aplicados à detecção de pedestres, criando um modelo baseado em *SVM* utilizando como extrator de características um histograma de características orientado, ou *HOG*, combinado com o *SIFT*.

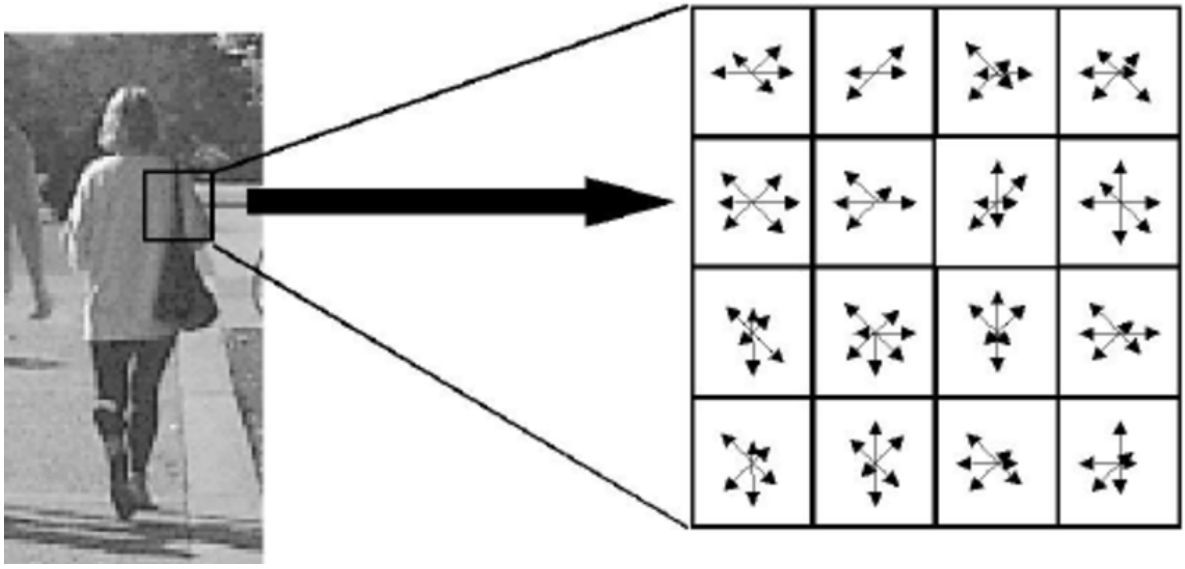


Figura 11 – Exemplo de *features* extraídas em (DALAL; TRIGGS, 2005)

Depois de analisarem os descritores existentes, baseados em gradiente e bordas, os autores demonstraram experimentalmente que *Grids* utilizando *HOG* superam significativamente os conjuntos de *features* até então existentes para detecção de pessoas em imagens, um exemplo pode ser visto na Figura 11. Nesse trabalho, os autores verificaram a influência de cada estágio de processamento no desempenho, concluindo que uma série de descritores sobrepostos são

importantes para bons resultados. A abordagem possibilitou uma separação de quase todas as pessoas presentes na *MIT pedestrian database*.

Recentemente, técnicas de aprendizagem profunda tornaram-se populares para realizar reconhecimento de objetos. Modelos como Redes Neurais Convolucionais, ou CNNs, começaram a ser utilizadas largamente para aprender automaticamente *features* de objetos e identificalos em diferentes setores. Isso foi possível com os avanços dos *hardwares* que, com o poder de analisar milhares de dados de treinamento, possibilitaram que os modelos de redes tornassem capazes de aprender características mais específicas das imagens, além de atigirem velocidade de processamento próxima ao tempo real.

Em um dos trabalhos mais importantes introduzidos nesta linha de pesquisa da história recente, os autores (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) utilizaram uma Rede Neural Convolucional (ou CNN) para detectar uma enorme quantidade de objetos sobre a base de imagens *ImageNet* (Seção 6.0.1). A pesquisa leva a concluir que, mesmo havendo variação na profundidade e amplitude da CNN, é possível realizar inferências corretas sobre a natureza das imagens. Além disso, verifica-se que, em comparação com Redes Neurais *FeedForward* padrão, com camadas de tamanho similar, as CNNs têm menos conexões e parâmetros e, portanto, são mais fáceis de serem treinadas, enquanto seus desempenhos são equivalentes. A CNN proposta divide as imagens em várias regiões e classifica cada região em suas possíveis classes, possibilitando assim detectar mais de um objeto em uma mesma imagem.

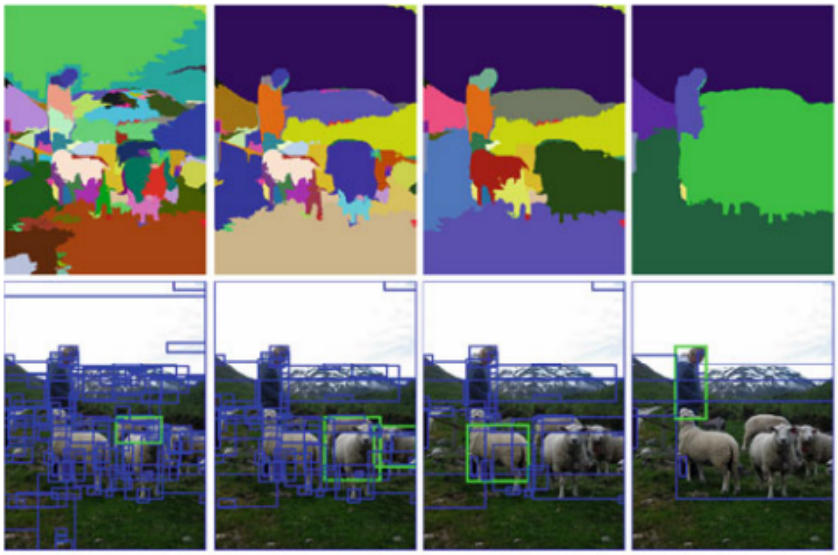


Figura 12 – Exemplo de seleção seletiva, reduzindo o número de testes realizados para buscar objetos de interesse. Proposto em (UIJLINGS et al., 2013)

Para aumentar a velocidade de execução das Redes Neurais Convolucionais, (UIJLINGS et al., 2013) propõe uma busca seletiva por áreas específicas da imagem, evitando assim uma

varredura por força bruta sobre cada imagem. É realizada uma combinação entre segmentação de imagens e busca por força bruta por objetos de interesse. Inicialmente, o algoritmo segmenta as imagens de entrada, e em seguida, nas regiões encontradas ele realiza a busca por objetos de interesse. Esse processo permite reduzir o número de sobreposições de *bounding boxes* que ocorrem durante o procedimento puramente exaustivo. Um exemplo do processo de segmentação e estimativa de *bounding boxes* pode ser visto na Figura 12. Mesmo com a redução do espaço de busca, tendo uma média de 10000 regiões verificadas, a sensibilidade do algoritmo foi de 99%, indicando sua alta assertividade.

Nos laboratórios da *Microsoft Research*, em (GIRSHICK et al., 2015), os autores iniciaram uma série de publicações evoluindo o método até atingir o estado da arte em detecção de objetos, aprimorando as Redes Neurais Convolucionais, nomeando o método de *R-CNN*. Eles implementaram uma busca seletiva para gerar regiões, extraindo cerca de 2000 regiões de cada imagem, com o tempo de processamento entre 40 e 50 segundos para cada imagem. As principais bases utilizadas neste método foram: (1) Aplicação de CNNs de forma *bottom-up* em regiões da imagem para localizar e segmentar objetos. (2) Para quando dados de treinamento supervisionado forem escassos, um pré-treinamento supervisionado, seguido por um ajuste fino específico do domínio, aumentam significativamente o desempenho do modelo. Um exemplo desta arquitetura pode ser vista na Figura 13.

Em (GIRSHICK, 2015), ainda nos laboratórios da *Microsoft Research*, os autores demonstram como aprimoraram ainda mais o algoritmo, aumentando sua velocidade de processamento, propondo o chamado *Fast-CNN*. Eles propõem um método de Rede Convolucional de Região Rápida (Fast R-CNN) para detecção de objetos. Em comparação com o trabalho anterior, o Fast R-CNN emprega várias inovações para melhorar a velocidade de treinamento e teste e, ao mesmo tempo, aumentar a precisão da detecção. O Fast R-CNN treina a rede VGG16 muito profunda 9 vezes mais rápida que a R-CNN. Cada imagem é passada apenas uma vez para que a CNN gere os mapas de *features*. A pesquisa seletiva é usada nesses mapas para gerar estimativas para cada objeto da imagem, levando o algoritmo a ser executado em 2 segundos por imagem.

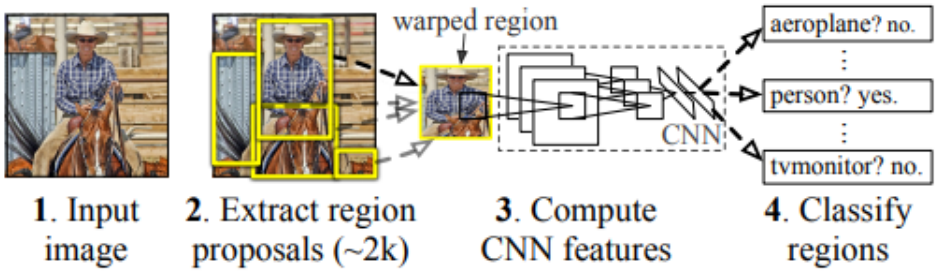


Figura 13 – Exemplo da arquitetura do R-CNN, retirado de (GIRSHICK et al., 2015)

Na sua última evolução relevante (REN et al., 2015), os autores demonstram mais um ponto de melhora no algoritmo, deixando-o ainda mais rápido, chamando-o agora de Faster R-CNN. Eles substituíram o método de pesquisa seletiva, utilizado na Fast R-CNN, pela rede de proposta de região (RPN), o que permite que o algoritmo seja executado a 5fps (0,2 segundos), levando o algoritmo a próximo do tempo real. O RPN é uma rede convolucional que infere *bounding boxes* (ou limites dos objetos), ao passo que estima a qual classe cada objeto pertence em uma imagem.

Em (HE et al., 2016), foi apresentada a *RetinaNet*, também conhecida como *ResNet*. Seus autores atingiram um marco muito relevante, onde na competição *ILSVRC 2015*, ultrapassaram o desempenho humano no reconhecimento de objetos para as imagens apresentadas. Sua taxa de erro ficou em 3.6%, enquanto o erro humano ficou calculado entre 5% e 10%. Os autores reformularam as camadas de redes neurais muito profundas, como funções de aprendizagem residuais com referência aos valores de entrada das camadas, ao invés de utilizarem funções de aprendizagem não referenciadas. Para isso, demonstraram evidências empíricas abrangentes para afirmar que tais redes residuais são mais fáceis de otimizar e podem ganhar precisão a partir do aumento considerável da profundidade. A *ResNet* empilha sequencialmente blocos residuais, onde cada um calcula a mudança residual que será aplicada aos seus valores iniciais, adicionando este resultado à sua entrada novamente, produzindo assim suas representações de saída.

Uma extensão da Faster R-CNN é proposta por (HE et al., 2017), com a Mask R-CNN. A nova abordagem detecta objetos de maneira eficiente em uma imagem, sobre máscaras de segmentação para cada parte separada da imagem, geradas em paralelo. A segmentação utiliza máscaras referentes a objetos das classes conhecidas, o que proporciona uma redução ainda maior do espaço de busca. Além disso, substitui o algoritmo de alinhamento utilizado originalmente (*RolPool*), que não consegue alinhar a nível de pixel as imagens entre as camadas da rede, pelo algoritmo *RolAlign*, que utiliza interpolação bi-linear para resolver o problema. O

novo método tem maior assertividade quando comparado ao Faster R-CNN, aumentado em 5.7 pontos *AP* na base de imagens Microsoft CoCo, mantendo a velocidade de processamento em 5fps.

Em (TANG et al., 2018), os autores trazem uma solução para imagens médicas volumétricas chamado de *Segmentation-by-Detection*. O método possui um módulo de detecção e outro de segmentação. O módulo de detecção encontra regiões de interesse e produz um conjunto de regiões que podem ser de atenção para o módulo de segmentação. O módulo de segmentação recebe uma imagem volumétrica como entrada e as suas regiões de interesse, direcionando a segmentação nestas regiões.

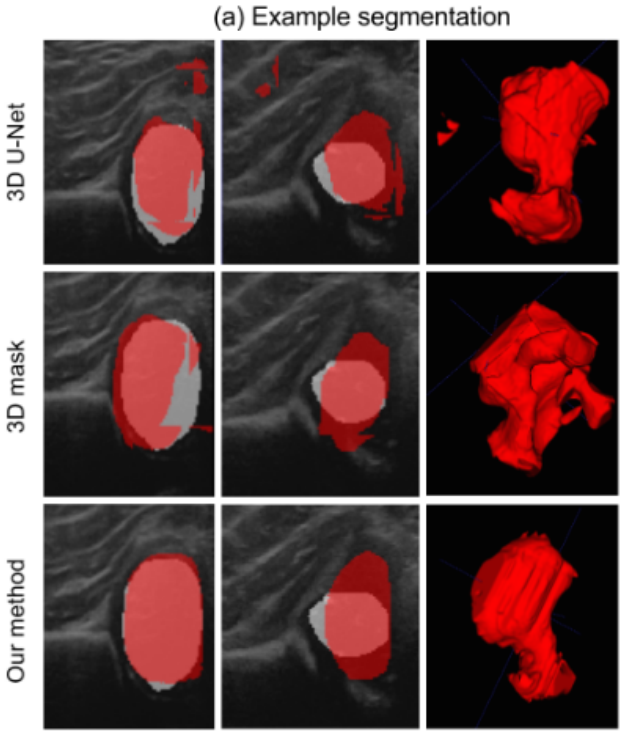


Figura 14 – Resultados do algoritmo *SEGMENTATION-BY-DETECTION* em uma imagem de cabeça femoral, retirado de (TANG et al., 2018)

Os resultados mostraram-se mais precisos e eficientes comparados com o método *3D U-Net* e o *3D mask*. Como visto na Figura 14, em (a), a segmentação automática está em vermelho enquanto o *ground truth* está em branco. As duas primeiras linhas mostram os resultados da *3D U-Net* e da *3D Mask* (métodos que apenas segmentam as regiões de interesse em volumes 3D), respectivamente, enquanto o método do autor é exibido na parte inferior. As duas primeiras colunas são os resultados de segmentação em regiões específicas da imagem, enquanto a terceira coluna exibe a vista 3D da cabeça femoral segmentada e identificada (com o *Segmentation-by-Detection*).

3.2.2 YOLO

Em (REDMON et al., 2016), uma abordagem para realizar reconhecimento de objetos foi apresentada. Buscando reduzir o tempo de detecção de objetos, um grupo de pesquisadores liderados por Joseph Redmon apresentaram um modelo de regressão sobre redes neurais convolucionais para solucionar o problema de reconhecimento e detecção de objetos em imagens de forma rápida, esse método foi batizado de *YOLO*.

O *YOLO* carrega em seu acrônomo sua principal característica, *you only look once* (em tradução literal, você apenas olha uma vez), ou seja, sua dinâmica de reconhecimento considera apenas uma fração dos frames de entrada a cada instante de tempo. Este comportamento foi proposto pelos autores ao identificarem que o sistema visual humano permite que ao olhar uma imagem, mesmo em uma fração de segundos, os objetos contidos na cena sejam identificados de forma rápida e objetiva (REDMON et al., 2016). A partir deste objetivo, os autores buscaram inicialmente criar uma solução que pudesse auxiliar no desenvolvimento de aplicações que necessitam de baixo tempo de processamento, como carros autônomos, que poderiam ser aprimorados para reduzir a utilização de sensores especiais, aproveitando-se da alta velocidade no reconhecimento dos objetos em cenas de trânsito, próxima a tempo real.

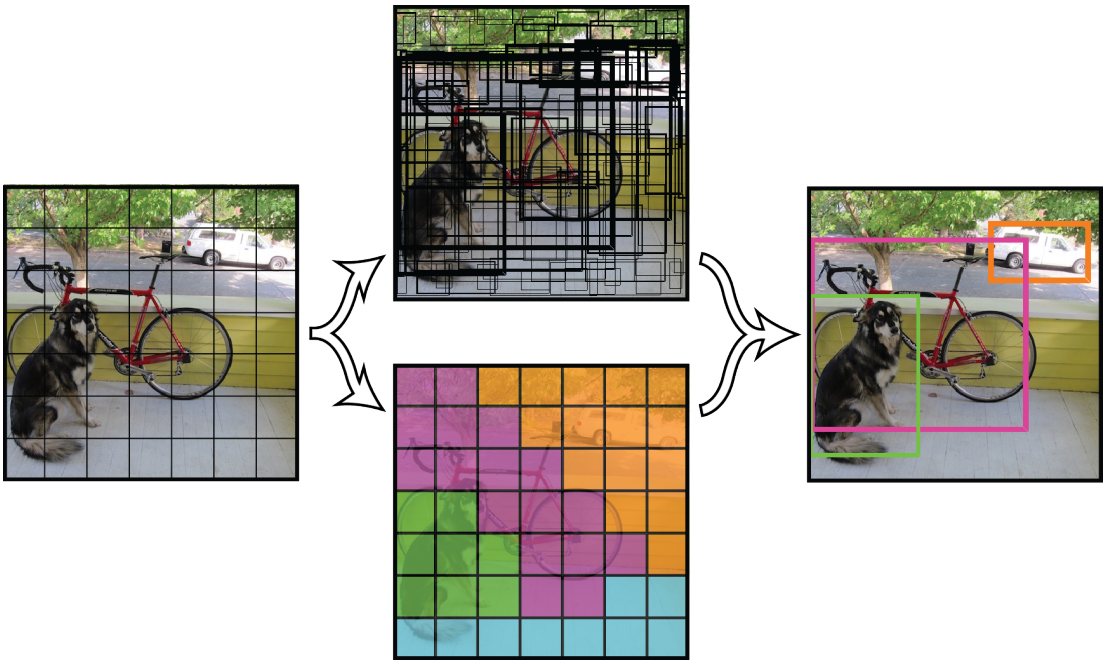


Figura 15 – Modelo de funcionamento do *YOLO*, adaptado de (REDMON et al., 2016)

Em sua primeira versão (REDMON et al., 2016), chamada de *YOLOv1*, introduziu um modelo unificado de detecção de objetos, onde uma única rede convolucional infere, simultaneamente, *bounding boxes* (revisado na Seção 2.1) e probabilidades para cada uma delas.

As probabilidades calculadas buscam estimar a qual classe de imagens pertence (préviamente conhecidas) cada segmento restringido pelas *bounding boxes* inferidas. Esta rede neural convolucional proposta utiliza características da imagem como um todo para inferir cada *bounding box* calculada. Além disso, calcula uma inferencia de todas as *bounding boxes* sobre todas as classes de imagens conhecidas de forma simultânea (REDMON et al., 2016).

Para as imagens de entrada, o método propõe dividir cada *frame* sobre um *Grid* SXS. Cada subdivisão deste *Grid* é chamada de célula, sendo essas responsáveis por detectar se um objeto de interesse (ou parte dele), está concentrado em sua região de demarcação. Na Figura (15), a imagem mais à esquerda representa este *Grid* que divide a imagem inicialmente.

Cada conjunto de células infere B *bounding boxes* e o nível de confiança de cada uma conter um objeto de interesse de uma determinada classe, além da acurácia estimada do objeto pertencer à cada uma das classes conhecidas, formalmente definida na Expressão (2) (REDMON et al., 2016).

$$P(Objeto) \times IOU_{ouro}^{padrão} \tag{2}$$

Se nenhum objeto existe na célula do *Grid*, o nível de confiança calculado deve ser 0. Caso exista um objeto na célula em questão, o valor da confiança será calculado pelo índice de similaridade IoU (revisado na Seção 2.1), entre a *bounding box* e o padrão-ouro (também conhecido como *Ground truth*).

Para cada uma das classes de imagens conhecidas, os *Grids* têm suas probabilidades de ocorrência calculadas. No fim, a imagem fica dividida em *Grids* $S \times$S, $onde$S$\in R$, com cada um calculando a probabilidade de B *bounding boxes* conterem o objeto de C classes de interesse, como reprensentado na Figura (15), na imagem inferior central. Os *Grids* são mantidos em um tensor, como o descrito na Expressão (3).

$$S \times S \times (B * 5 + C) \tag{3}$$

O *YOLO* prevê várias caixas delimitadoras por *Grid* de células. Para calcular a função de perda para resultados *True Positives, TP*, apenas uma das *bounding boxes* deve ser responsável pelo objeto. Para este propósito, é selecionada aquela com a mais alta IoU em relação ao padrão-ouro. Essa estratégia tende a melhorar as inferencias das *bounding boxes*, para previsões espaciais das classes de objetos (REDMON et al., 2016). Na *Yolo*, a função de penalização (Equação 4) é composta pela perda na classificação, perda na localização (erros entre a *boun-*

ding box inferida e o padrão-ouro), além da perda no valor de confiança do objeto pertencer a determinada classe.

$$\sum_{i=0}^{48}\left(\lambda 1_i^{\text{obj}}\left((x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2+(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2\right)+\sum_{c\in\text{classes}}(p_i(c)-\hat{p}_i(c))^2\right) \quad (4)$$

Na Equação (4), 1_i^{obj} simboliza cada objeto que aparece na célula i. Caso não exista um objeto na célula, nenhum valor de penalização é aplicado para a *bounding box*, pois neste caso não há padrão-ouro para ser comparado, então são penalizadas as probabilidades associadas a essa região (REDMON et al., 2016).

A rede é treinada por 135 períodos, com 21 mil imagens no total. Durante o treinamento o tamanho do *batch* utilizado foi de 64, um momento de 0.9, com decaimento de 0.0005. São utilizadas para o primeiro período, uma taxa de aprendizagem de 10^{-2}. A taxa de aprendizagem permanece em 10^{-2} por 75 períodos, quando então é ajustada para 10^{-3} por 30 períodos, e por fim, para 10^{-4} por mais 30 períodos. Para evitar *overfitting*, uma variação é aplicada sobre os dados, sendo a uma taxa de = .5 após a primeira camada conectada, assim impedindo a co-adaptação entre as camadas da rede (REDMON et al., 2016).

O *YOLOv1* foi desenvolvido sobre a *framework DarkNet*, sendo treinada no banco de dados ImageNet-1000 (vista na Seção 6.0.1). A *framework DarkNet* foi modificada para realizar detecção de objetos, tendo adicionada 4 camadas convolucionais e 2 camadas totalmente conectadas nos últimos níveis. Essa arquitetura pode ser considerada simples quando comparada com detectores complexos de dois estágios, como o Faster-RCNN (REN et al., 2015). Nesta primeira geração, a rede combinada possui 24 camadas convolucionais no total, sendo 2 camadas totalmente conectadas (REDMON et al., 2016).

Em sua primeira evolução (REDMON; FARHADI, 2017), uma nova arquitetura da Darknet foi utilizada, agora possuindo 19 camadas convolucionais e 5 camadas totalmente conectadas no topo do último nível convolucional, sendo chamada de *Yolo9000*. Foi adicionado uma Normalização em *Batch* para a todas as camadas convolucionai, técnica utilizada para normalizar a camada de entrada, ajustando e dimensionando as ativações, o que melhorou os resultados medidos por mAP em 2%.

Uma outra melhoria trazida em (REDMON; FARHADI, 2017), foi o aumento da resolução utilizada nas imagens de entrada na fase de treinamento, que passaram de 256×256, para 448×448. Desde a *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), a resolução padrão utilizada por pesquisadores era baixa próximas do mesmo, ao aumentarem a resolução

das imagens, os resultados mAP alcançados também elevaram-se, obtendo precisões cerca de 4% melhores.

Na sua última evolução, a principal mudança se deu pela alteração da função de penalização em (REDMON; FARHADI, 2018). Os três últimos termos da função que penaliza o processo de reconhecimento na *Yolo9000* são erros quadrados, enquanto no *YOLOv3* eles foram substituídos por termos de erro de entropia cruzada.Portanto, a confiança objetiva e as previsões das classes que são inferidas no *YOLOv3* agora são calculadas por meio de uma regressão logística. Enquanto o detector é treinado, para cada padrão-ouro de *bounding boxes*, é atribuida uma caixa delimitadora, cujo deve conter valor máximo de sobreposição entre ela e o padrão-ouro.

4 LIDA

O *Learning Intelligent Distribution Agent*, ou *Learning* IDA (LIDA), é um modelo cognitivo computacional apresentado como uma evolução do modelo IDA (Seção 3.1.4), onde foram adicionados três tipos de aprendizagem: perceptiva, episódica e procedural. Conforme sua evolução, o modelo se tornou genérico, agregando diferentes aplicações, como visto em (FRANKLIN; JR, 2006), (FRANKLIN et al., 2007), (FRANKLIN et al., 2014) e (FRANKLIN et al., 2016). O último, (FRANKLIN et al., 2016), apresenta o modelo LIDA como um modelo cognitivo conceitual e parcialmente computacional, uma vez que nem todos os seus módulos conceituais ainda foram implementados. Esta arquitetura tem como objetivo modelar mentes, não cérebros.

(FRANKLIN et al., 2016) ainda demonstra como o modelo é dividido em módulos que atuam para que o agente possa perceber o ambiente e agir de acordo com seus próprios objetivos. Quando tais módulos trabalham em conjunto, é chamado de átomo cognitivo, onde são realizados diferentes processos cognitivos de alto nível. Porém, para que seja possível agir no ambiente apenas um átomo cognitivo não é o suficiente, visto que, existem tarefas complexas que necessitam de diferentes processos cognitivos para serem realizadas. Portanto, tais átomos são realizados continuamente no Ciclo Cognitivo do LIDA, que será explicado a seguir.

Do mesmo modo, serão apresentados as definições de seus módulos, os comprometimentos que o modelo tem com a neurociência, um exemplo de um agente em desenvolvimento e o *framework* que auxilia no desenvolvimento de agentes.

4.1 CICLO COGNITIVO DO LIDA

Para o LIDA, o ciclo cognitivo é um ciclo que permite amostragens e respostas frequentes. Pode ser visto tal qual um átomo cognitivo onde são realizados processos cognitivos de alto nível, como planejamento, resolução de problemas, etc. A Figura 16 mostra as três fases que esse ciclo é dividido: a fase de percepção e entendimento, a fase de atenção e a fase de ação e aprendizado. A primeira fase usa dados externos para entender a situação atual; a segunda filtra o conteúdo de acordo com seu grau de relevância e propaga globalmente o conteúdo tido como mais relevante; por fim a terceira fase seleciona uma resposta e a executa, assim como aprende com os sistemas de memória.

Este ciclo, igualmente ao próprio modelo LIDA, é descrito por módulos que, apesar de terem suas funções bem definidas, possuem grande interação entre si. É importante notar que os processos no LIDA ocorrem paralelamente, com exceção do processo de consciência e da seleção de ação. Além disso, podem ser divididos em três categorias: nunca conscientes, pré-conscientes e conscientes. Seus sistemas de memória variam entre curto prazo e longo prazo.

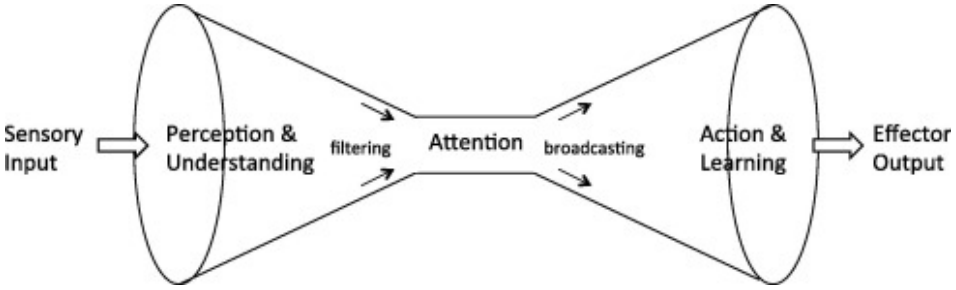


Figura 16 – Diagrama esquemático do ciclo cognitivo do modelo LIDA (FRANKLIN et al., 2016).

A fase de Percepção e Entendimento inicia a partir de estímulos externos e internos enviados à *Sensory Memory* (Figura 17). O conteúdo resultante deste gera percepção ao passar pela *Perceptual Associative Memory* e é disponibilizada no *Current Situational Model*. O *Current Situational Model* que mantem-se atualizado alimentando a *Perceptual Associative Memory*, a *Spatial Memory*, a *Transient Episodic Memory* e a *Declarative Memory*. De mesmo modo, as atualizações no *Workspace* são feitas pela *Structure Building Contents* utilizando os dados provindos do *Current Situational Model* e da *Conscious Contents Queue*, gerando o que é chamado de estado de pré-consciência no agente.

Durante a fase de atenção, os *Attention Codelets*[1] avaliam o *Current Situational Model* em busca de selecionar conteúdo para ser levado ao estado de consciência. Ao encontrar tal conteúdo, então, é criado uma aliança para competir pelo acesso à consciência e, deste modo, a aliança vencedora terá seu conteúdo propagado globalmente. Essas alianças são geradas, pois, os *Codelets* de Atenção podem colocar mais que uma estrutura nelas, assim como diferentes *Codelets* de Atenção podem combinar suas estruturas resultantes a fim de criar uma aliança em conjunto.

Na fase de Ação e Aprendizado, grande parte dos módulos do LIDA selecionam conteúdos do processo de consciência que são adequados ao seu aprendizado, assim como selecionam comportamentos guardados na *Procedural Memory* que respondam corretamente aos estímulos de entrada e só então o módulo *Action Selection* irá selecionar um comportamento para ser en-

[1] *Codelet* é um pequeno processo que procura continuamente condições para agir em busca de uma atividade específica.

viado ao *Sensory Motot Plan*. Este, por sua vez, cria ou seleciona um plano motor adequado para ser executado. De modo a finalizar o ciclo cognitivo do LIDA.

4.2 MÓDULOS DO AGENTE E SUAS FUNÇÕES

Cada módulo citado na seção 4.1 tem sua função definida e delimitada, estes módulos serão descritos a seguir e podem ser vistos na Figura 17.

A Memória Sensorial (Quadro *Sensory Memory* da Figura 17) é responsável por receber estímulos dos sensores do agente, como câmeras ou microfones, transformá-los em representações e armazená-los por um curto período de tempo.

Essas representações são processadas por detectores de características e, então, enviadas à Memória Perceptiva. Sua implementação ainda está em aberto, duas ideias correntes são usar *deep learning*, que possui uma performance promissora para reconhecimento visual, e implementá-la como uma Memória Hierárquica Temporal, utilizando Algoritmos de Aprendizado Corticais.

A Memória Perceptiva Associativa (Quadro *Perceptual Associative Memory* da Figura 17), também chamada de PAM, é um modelo de memória de longo prazo onde são representadas informações sensoriais usando nós e *links*. Os nós representam detectores de características, como objetos e ações, e os *links* representando relações entre os nós, como causa e inibição. A memória perceptiva pode ser ativada por estruturas vindas tanto da memória sensorial como do modelo situacional corrente.

Por sua vez, a Memória Espacial (Quadro *Spacial Memory* da Figura 17) é o sistema de memória que codifica e guarda informações espaciais do ambiente e da própria orientação do agente. As representações espacias do LIDA são construídas no *Workspace* a fim de identificar entidades conhecidas, PAM, e suas posições no ambiente.

Posições essas representadas por links especias do PAM com informações de posição e, por sua vez, esses links são representados como vetores espaciais concêntricos entre a posição do agente e a do objeto. (FRANKLIN et al., 2016) mostra o uso do *Grid* Espacial Alocêntrico como uma forma de se implementar a Memória Espacial. Ainda demonstra que o *Grid* Espacial Alocêntrico é um *grid* PAM, onde os nós representam lugares específicos no ambiente que podem ser conectados a objetos conhecidos e é atualizado durante movimento do agente.

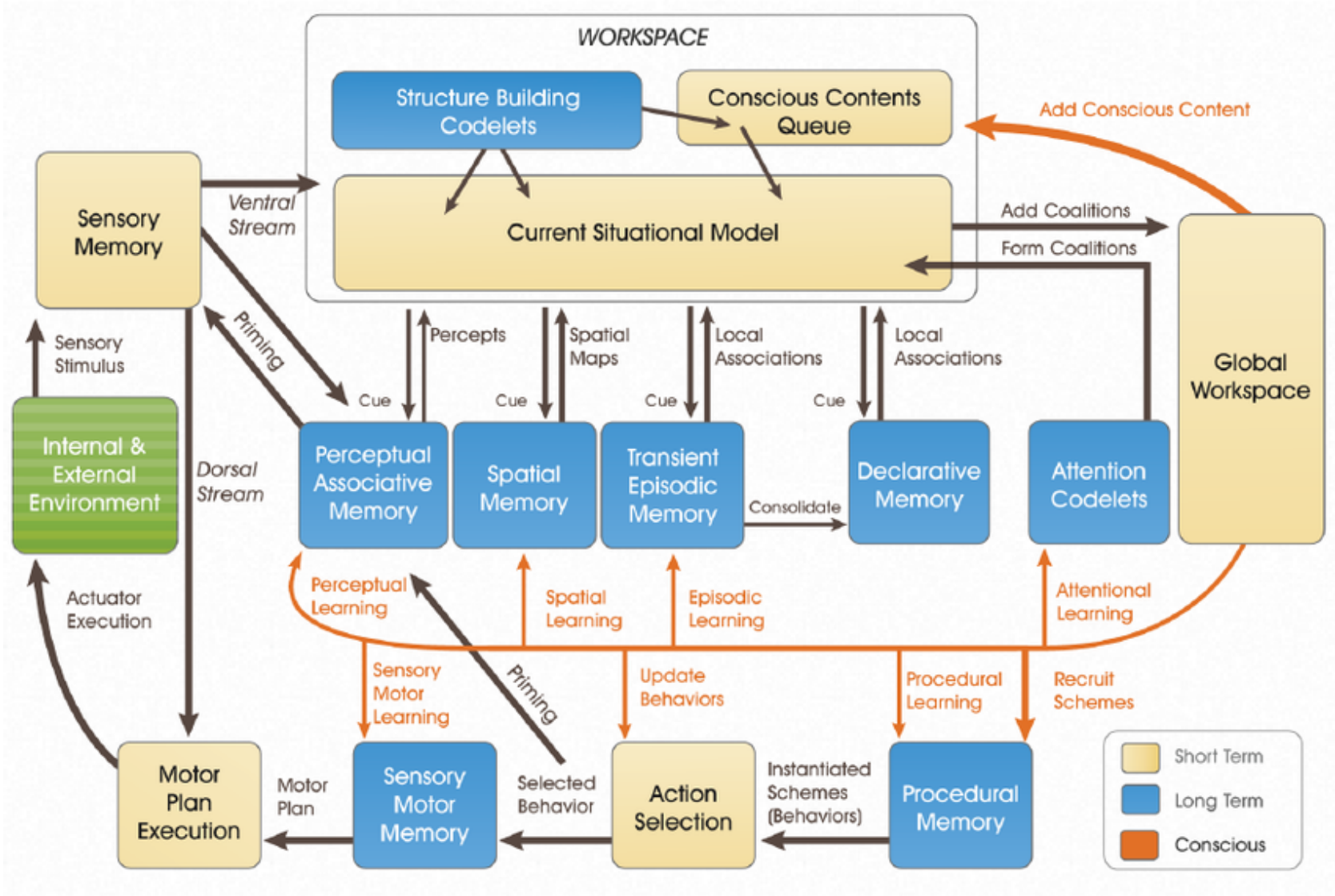


Figura 17 – Ciclo cognitivo do modelo LIDA (FRANKLIN et al., 2016).

A Memória Episódica é armazenadora de eventos que representam processos que respondem por informações situacionais (o que?), temporais (quando?) e de localização (onde?), geralmente informações de longo prazo. Entretanto, o modelo LIDA utiliza uma versão mais simples dessa memória chamada de Memória Episódica Transiente (Quadro *Transient Episodic Memory* da Figura 17), onde os eventos decaem com o tempo quando não há um reforço em sua ativação, de modo que elas não duram mais que algumas horas ou um dia. Em algum momento, quando o agente fica offline, as memórias que não decaíram são consolidadas na Memória Declarativa. Essa consolidação cria novos traços de memória para novos eventos e reforça traços já existentes.

A Memória Declarativa (Quadro *Declarative Memory* da Figura 17) é uma memória de longo prazo que não recebe informações de propagação consciente e sim por consolidação, como visto acima. Nela existem dois tipos de memória, a Memória Autobiográfica e a Memória Semântica. A Memória Autobiográfica é o tipo de memória de eventos completos, possuindo as informações situacionais, temporais e de localização sem perdas. Por sua vez, a Memória Semântica armazena memórias com perda de traços temporais e de localização, desse modo o que restar da memória é mantido na forma de fatos, regras e etc.

O *Workspace* (Quadro *Workspace* da Figura 17) pré-consciente é uma memória de curto prazo com uma latência de décimos de segundos, que é pré-consciente por suas representações não serem conscientes. Ela é formada pelos módulos de Modelo Situacional Atual e pela Fila de Conteúdo Consciente, assim como pelos *Codelets* Construtores de Estruturas que processam os conteúdos dos outros dois.

O Modelo Modelo Situacional Atual (Quadro *Current Situational Model* da Figura 17), CSM, mantem informações continuamente atualizadas pelos dados gerados tanto pelo PAM quanto pela Memória Sensorial para que ele esteja sempre a par da situação atual do agente. Entretanto, o CSM pode armazenar também estruturas mais complexas, como planos. Essas estruturas chegam a ele via memórias de longo prazo. Sempre que uma estrutura nova chega ao CSM, é enviada às memórias de longo prazo para que sejam geradas novas associações locais. Essas associações são enviadas ao CSM, e, então, são enviadas às memórias novamente para que possam gerar novas associações. Portanto, o CSM é sempre atualizado ao longo do tempo. Entretanto suas estruturas decaem rapidamente durando décimos de segundos.

As estruturas do CSM também são criadas pelos *Codelets* Construtores de Estruturas (Quadro *Structure Building Codelets* da Figura 17). Esses *codelets* são processos que reconhecem relações entre conceitos e objetos, como similaridade ou casualidade. Para realizar isso, eles monitoram a CSM e a Fila de Conteúdo Consciente afim encontrar conteúdos de seu inte-

resse. Quando esse conteúdo é encontrado, aplica-se uma ação que resulta em modificações no CSM. Sempre que uma nova estrutura é criada, deve-se, então, adicionar funções de ativação a elas, utilizando como base a ativação das estruturas usadas para criá-las.

A Fila de Conteúdo Consciente (Quadro *Councious Contents Queue* da Figura 17) é uma fila de conteúdos que foram propagados conscientemente nos últimos décimos de segundos, tornando-a de muito curto prazo. Sempre que um conteúdo é propagado pelo módulo de consciência, entra no final da fila, tirando o conteúdo que esta à frente. Porém, os *codelets* construtores de conteúdos podem utilizar qualquer conteúdo da fila, não somente o que está à frente.

No modelo LIDA a atenção age como um filtro de relevância, onde os conteúdos mais relevantes são selecionados e enviados ao *Global Workspace*, que será explicado mais adiante. Para isso, são usados os *Codelets* de Atenção (Quadro *Attention Codelets* da Figura 17), esses *codelets* buscam conteúdos relevantes, seguindo um conceito de relevância próprio. Um exemplo do significado de relevância pode ser visto no *codelet* de expectativa, para ele um conteúdo relevante é aquele conteúdo esperado como resposta à uma execução de ação própria. Ao achar algum conteúdo relevante, os *codelets* incorporam esse conteúdo em uma aliança e atribuem a ela um grau de ativação, tal ativação irá servir como base para a competição do *Global Workspace*. Existem quatro tipos de *codelet* de atenção, são eles: *Codelet* de Atenção Padrão, que seleciona a estrutura mais relevante no CSM pelo seu grau de ativação; *Codelet* de Atenção Específica que busca conteúdos específicos que foram aprendidos por eles; *Codelets* de Expectativa, onde são buscados resultados de suas próprias ações; e os *Codelets* de Intenção, onde são buscados conteúdos que podem ajudar o agente a atingir seu objetivo.

É no *Global Workspace* (Quadro *Global Workspace* da Figura 17) que as alianças geradas pelos *codelets* de atenção competem pela propagação global, chamada também de propagação consciente. Nesse caso, ganha a competição quem possuir maior nível de ativação. Porém, essas competições não são assíncronas nem operam com *clock*, é necessário de algo que inicie a competição. Assim, foram usados quatro gatilhos diferentes. O primeiro é um simples limite de ativação, ao chegar uma nova aliança que possua um nível de ativação acima de um determinado limite uma competição é iniciada. Outro gatilho é acionado no momento em que a soma de ativações das alianças presentes no *Global Workspace* excede um determinado limite. Pode-se iniciar uma competição também caso nenhuma aliança chegue ao *Global Workspace* por um determinado limite de tempo e, por fim, existe o gatilho padrão, onde uma competição é iniciada após passar um período de tempo sem que tenha ocorrido uma propagação consciente.

Quando uma propagação consciente é feita, um dos módulos de memória que recebem essa informação é a Memória Procedural (Quadro *Procedural Model* da Figura 17). Ela armazena informações do que deve ser feito em determinada situação para que seja possível chegar a um objetivo. A sua estrutura básica é o *scheme* que consiste em um contexto, uma ação, um resultado e um nível de ativação base. Sempre que essa memória recebe uma propagação consciente, todo *scheme* cujo contexto da sequência ao conteúdo da propagação é, então, instanciado, torna-se um comportamento e é enviado ao módulo de seleção de ação.

Ao chegar na Seleção de Ação (Quadro *Action Seletion* da Figura 17), os comportamentos instanciados, passam a ser nós de uma rede de comportamento. Esta rede é uma forma melhorada das *behaviors nets* apresentadas em (MAES, 1989). Por ser um dos poucos módulos assíncronos, a Seleção de Ação necessita de um esquema de deliberação especial para decidir quando deve-se iniciar o processo. Existem 3 tipos de gatilhos para iniciar esse processo: a ativação de um comportamento estar acima de um determinado nível, a ativação total de todos os comportamentos na rede de Seleção de Ação estar acima de determinado nível e quando nenhuma ação é executada em determinada quantidade de tempo. Quando o processo de deliberação é iniciado o comportamento que melhor se encaixar no contexto atual é selecionado e, então, enviado para a PAM, caso seja um comportamento interno, ou para a Memória Sensorial Motora, caso seja externo. O envio do comportamento a PAM é chamado de *Priming*.

A Memória Sensorial Motora (Quadro *Sensory Motor Memory* da Figura 17), por sua vez, seleciona um plano motor que realize esse comportamento e o envia para o Executor de Plano Motor, para que ele seja devidamente posto em prática. O plano motor é uma estrutura de dados baseada na arquitetura de subsunção, onde os dados sensoriais são conectados diretamente com comandos motores. Os comandos gerados são enviados ao Executor de Plano Motor (Quadro *Motor Plan Execution* da Figura 17), para que o agente possa agir no ambiente de acordo com o que foi percebido no início do ciclo cognitivo.

Grande parte desses módulos possuem algum tipo de aprendizado, esse aprendizado utiliza a propagação consciente como principal meio de obter informações novas. Essa propagação global permite que a PAM aprenda novas entidades, assim como reforçar as ativações daquelas existentes. No caso da Memória Episódica Transiente, aqueles que são aprendido e tem suas ativações reforçadas são os eventos. A Memória Espacial, por sua vez, utiliza a propagação consciente para aprender representações alocêntricas. Os *codelets* de atenção, porém, apenas reforçam seu grau de ativação, a utilização de conteúdo da propagação consciente para gerar novos *codelets* ainda é apenas teoria.

Por sua vez a Memória Procedural possui dois tipos de aprendizado, selecionista e instrucionista. No aprendizado selecionista todo *scheme* que é executado recebe um reforço em sua ativação. No instrucionista a propagação consciente sugere a criação de um novo *scheme* a partir de um antigo, adicionando ou removendo estruturas dele. Esse novo *scheme* utiliza a ativação do antigo e a propagação consciente para determinar sua própria ativação, de forma que o antigo *scheme* ainda exista na Memória Procedural.

Por fim, o módulo de Seleção de Ação é um pouco diferente, pois existe apenas uma atualização da ativação de seus *schemes*, uma vez que os novos provêm da Memória Procedural. No momento em que uma ação é selecionada, um *codelet* de expectativa espera que os resultados dessa ação apareçam no CSM. Caso esses resultados apareçam e sejam os esperados, então, a ativação de quem gerou a ação é reforçada, porém, se a ação gerar resultados indesejados, ela recebe uma atenuação em sua ativação.

4.3 COMPROMENTIMENTOS DO MODELO

O modelo LIDA tem em suas bases diretrizes do que busca implementar. Por não ser um modelo que busca simular biológicamente um encéfalo humano, seu escopo foi maturado e definido a partir de regras direcionais da forma listada a seguir:

a) Modelagem de um sistema de nível cognitivo: grande parte dos modelos são restritos a alguma função da cognição, portanto relacionar esses modelos limitados é algo que gera grande dificuldade. O LIDA é um modelo computacional que busca unificar esses modelos com o intuito de ser responsável pela percepção de estímulos externos, pelo processamento coerente resultante, culminando em seleção e execução de uma ação.

b) Cognição incorporada e situada: cognição incorporada afirma que tanto o corpo quanto o cérebro afetam o processamento cognitivo, enquanto a cognição situada defende a influência do ambiente nesse processo. O LIDA respeita ambas as implicações ao usar símbolos perceptivos e evitando símbolos amodais, aqueles que não possuem estados, sem contexto e isolados. Ainda é possível ver estas situações na interação contínua e mútua com o ambiente, com o papel ativo dos processos internos e na falta de separação entre percepção e ação.

c) Ciclo Cognitivo com átomo cognitivo: cada ciclo cognitivo executa uma resposta utilizando conteúdos conscientes e processamento pré-consciente. Ações de alto nível requerem uma sequência de ações, portanto devem ser implementadas por ciclos múltiplos.

d) *Global WorkSpace Theory* (GWT): o modelo LIDA apresenta atenção segundo o que é descrito pelo GWT, onde ela é descrita como o processo de trazer conteúdo relevante para a consciência, o que pode ser visto na fase de atenção de seu ciclo cognitivo. Assim como apresenta alianças geradas por processos especializados, que competem pela consciência, onde aquele tido como vencedor é, então, propagado e usado tanto para a Seleção de Ação como no aprendizado.

e) Aprendizado Consciente: o modelo LIDA da suporte à Hipótese de Aprendizado Consciente, que diz que uma parte significante do aprendizado ocorre na interação da consciência com os sistemas de memória.

f) Decaimento compreensível das representações e memórias: os módulos do LIDA baseiam-se em processos gerados em representações estruturadas. Tais representações têm o dígrafo como dado fundamental, onde a partir dele são desenvolvidas estruturas mais complexas. Essas estruturas complexas possuem atributos associados a elas que decaem com o tempo. Ao atingir um limiar, elas podem ser retiradas do *workspace*, por exemplo. Esse comprometimento do LIDA está de acordo com um dos quatro requeridos para se ter um sistema auto-organizável, onde é dito que o sistema deve ser dissipativo.

g) Abundância de Aprendizado: o aprendizado no LIDA é desregulado, acontece em qualquer sistema e a qualquer momento. O agente deve aprender tanto de maneira instrucionista, onde novas entidades são representadas, quanto de modo selecionista, reforço da ativação do nível base ou variável apropriada.

h) Sentimentos são motivadores e moduladores de aprendizado: o LIDA modela os sentimentos tanto como motivadores quanto moduladores de aprendizado. Como motivadores, os sentimentos auxiliam na Seleção de Ação ao permitir uma rápida avaliação da situação. Por sua vez, como moduladores, eles auxiliam a regular a taxa de aprendizado. Eles são representados na Memória Perceptiva Associativa como nós.

i) Assincronismo: todos os processos dos módulos do modelo LIDA trabalham de forma independente e assíncrona, exceto a consciência e a seleção de ação. Essas, devem trabalhar de forma sequencial para que haja uma coerência nas ações do agente.

j) Memória Episódica Transiente: o modelo propõe uma memória onde são guardados detalhes sobre o que, onde e quando; que decaem ao longo de algumas horas ou um dia.

k) Consolidação: as memórias que foram armazenadas na Memória Episódica Transiente e que, ao final do dia, permanecem nela, são então consolidadas na Memória Declarativa.

l) Ponte não linear dinâmica à neurociência: apesar do modelo LIDA ter a intenção de modelar mentes, ainda assim deve ser consistente com evidências neurocientíficas conhecidas. A lacuna entre as representações cognitivas e a neurodinâmica subjacente pode ser preenchida por dinâmicas não lineares que exibem auto-organização. Entretanto, até então, não há implementação conhecida que apresente tal organização em seus processos.

4.4 FRAMEWORK LIDA

Com o intuito de facilitar o desenvolvimento de agentes LIDA, Stan Franklin demonstra em (FRANKLIN et al., 2016) o *Framework* LIDA. Este *framework* é escrito na linguagem de programação Java e implementa grande parte das funcionalidades de baixo nível necessárias pelos agentes LIDA, dentre essas funcionalidades estão: inicialização, assincronismo e gerenciamento de tarefas concorrentes, e criação de objetos. A Tabela 1 demonstra o nível de implementação de cada módulo.

Apesar das funcionalidades já estarem implementadas, ainda é necessário que se configure cada uma delas. Devido a esse fato, o *framework* possui um arquivo XML chamado *Agent Declaration File*. Nele os detalhes de configuração são feitos de forma declarativa, o que permite que se desenvolva um agente sem ter que modificar seu código. Entretanto, caso haja a necessidade de adicionar alguma funcionalidade diferente do padrão, o desenvolvedor deve implementar sua própria função e substituir a padrão pela sua neste arquivo de declaração.

Como visto durante este capítulo, grande parte dos processos do Agente LIDA é processado de forma assíncrona e busca simular a mente humana. Para que isso seja possível, a execução de cada tarefa deve ter um limite de tempo para ser realizada e, por essa razão, existe uma implementação padrão do *scheduler* de tarefas. Isto permite que sejam realizadas simulações de comportamento e experimentos na área da neurociência.

Para cada bloco apresentado na Figura (17) são implementados módulos no *framework*. Por exemplo, *Sensory Memory, Workspace* e *Action Selection* são módulos. Todos módulos possuem uma API em comum, mas também possuem sua própria API a qual define sua funcionalidade específica. Além disso, um módulo pode ter submódulos, como por exemplo o *Workspace* que tem como submódulos *Structure Building Codelets*, *Conscious Contents Queue*

e *Current Situational Model*. Os módulos são independentes tendo uma implementação padrão e podem ser estendidos por implementações do próprio usuário de um determinado domínio. Versões de cada módulo também podem ser implementadas pelo usuário sendo a compatibilidade garantida com outros módulos por meio das APIs em comum.

A comunicação entre módulos é feita seguindo o Design *Pattern Observer*. Um módulo "ouvinte"se registra como ouvinte em um módulo "produtor"; assim, então sempre que um produtor enviar algo, esse envio será para todos os ouvintes registrados nele. Entretanto, um mesmo módulo pode ser registrado tanto como "ouvinte"quanto como "produtor"em outros módulos, assim como pode ser ambos.

O *framework* utiliza uma estrutura de dados chamada *NodeStructure*, sendo essa um grafo com nós e *links* entre eles. É empregada na maioria dos módulos, os quais a utiliza para representar seus dados internos sendo uma espécie de “moeda comum” de representação entre muitos módulos. Os grafos são utilizados para representação conceitual de objetos, ações e eventos, e para a representação básica de dados no modelo LIDA. Além de grafo, são utilizadas outras estruturas como: para o módulo da *Transient Epsodic Memory* utiliza-se vetores de bits, *schemes* na *Procedural Memory*, alianças para o *Global Workspace* e comportamentos no *Action Selection*.

Neste trabalho, os seguintes módulos da Tabela 1 serão utilizados segundo proposições descritas no Capítulo 5, sendo eles: Memória Sensorial, *Global Workspace*, Memória Perceptiva Associativa, Memória Episódica Transiente, *Codelets* de Atenção e Memória Sensorial.

Tabela 1 – Módulos do LIDA e o estado de suas implementações (F - totalmente implementado, P - parcialmente implementado. N - ainda não implementado) (FRANKLIN et al., 2016).

Módulo	Framework LIDA	Agentes LIDA
Memória Sensorial	P	P
Memória Perceptiva Associativa	P	P
Codelets Construtores de Estruturas	P	P
Fila de Conteúdo Consciente	F	F
Workspace	F	F
Memória Espacial	N	P
Memória Episódica Transiente	F	F
Memória Declarativa	F	F
Codelets de Atenção	P	P
Global Workspace	F	F
Memória Procedural	F	F
Seleção de Ação	F	F
Memória Sensorial Motora	P	P
Executor de Plano Motor	P	P
Interface para robôs	N	P
Emoções, Avaliação	N	N
Aprendizado	P	P
Alarmes	N	N
Tomada de Decisão Volitiva	N	N
Tomada de Decisão Moral	N	P

5 METODOLOGIA

Neste trabalho, é proposta uma nova arquitetura para modelos de reconhecimento de objetos baseado em modelos cognitivos sobre redes neurais convolucionais. Esta nova arquitetura é sugerida sobre os conceitos neuro-cognitivos já estabelecidos pelo modelo *LIDA (Learning Intelligent Distribution Agent)* revisado na Seção 4.

Para a realização do reconhecimento de objetos nas imagens, o algoritmo *YOLO* (Seção 3.2.2) foi utilizado em sua versão 3, com a rede completa treinada e disponibilizada pelo seu autor. Esta versão já foi estendida e melhorada para reduzir seus principais problemas desde sua primeira implementação (REDMON; FARHADI, 2018). Para este trabalho, a nova arquitetura proposta prevê manter as características gerais do *YOLO*, porém dentro de uma arquitetura baseada no *LIDA*.

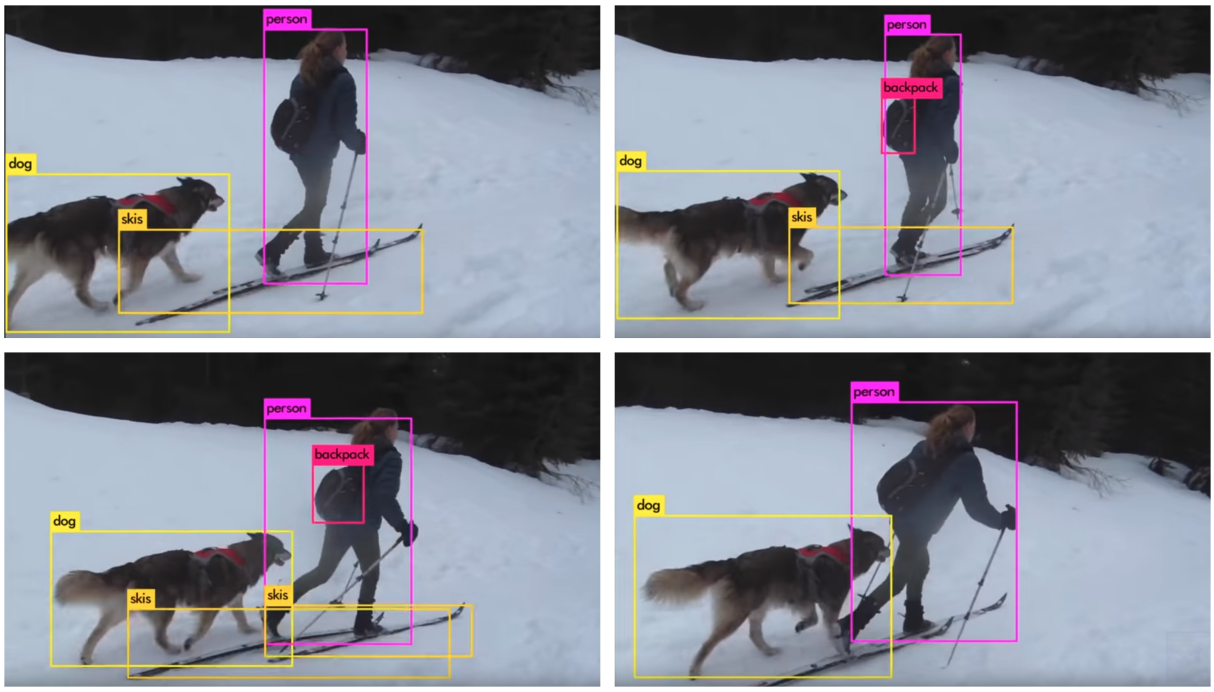


Figura 18 – Cena de 3 segundos com o *YOLO* realizando detecção de objetos e cometendo algumas falhas. No primeiro *frame* (esquerda superior) três objetos são detectados (cachorro, pessoa e equis de neve). Na segunda imagem (direita superior), uma mochila é detectada. Na terceira imagem (esquerda inferior), mais um esqui é detectado. No quarto *frame* (direita inferior), os esquis e a mochila deixam de serem detectados, mesmo permanecendo na mesma região da imagem.

Como visto na Seção 3.2.2, o *YOLO* carrega em seu acrônomo sua principal característica, *you only look once* (em tradução literal, você apenas olha uma vez), ou seja, sua dinâmica

de reconhecimento considera que todos os quadros de um video de entrada devem passar pela rede treinada, para que todos os objetos sejam reconhecidos todas as vezes.

Porém, o processo de detecção de objetos realizado pelo *YOLO* não pode ser considerado como cognitivo, uma vez que o encéfalo humano não realiza o reconhecimento dos objetos visíveis a todo momento (PELLI; TILLMAN, 2008). Além disso, o *YOLO* ainda não possui o melhor desempenho na fase de detecção dos objetos em cena dentre os algoritmos de reconhecimento da literatura recente (REDMON; FARHADI, 2018), tendo melhor assertividade apenas com objetos encontrados no *background* da imagem.

Com isso, este trabalho propõe uma evolução para o modelo de funcionamento do *YOLO*, buscando uma dinâmica cognitiva. Para isto, foi proposto o uso de módulos implementáveis da arquitetura *LIDA*, como o módulo de memória episódica transiente (Seção 4), com o objetivo de acompanhar regiões previamente detectadas em uma mesma cena. Desta forma é possível reduzir o tempo de detecção de objetos, além de aumentar a assertividade do modelo, sem a necessidade de submeter todos os quadros de um video ao processo de detecção realizado pela rede neural. Erros como o visto na Figura 18 são comuns, pois como a caracteristica principal do *YOLO* é realizar um reconhecimento a cada instante de tempo, detecções anteriores da mesma cena não possuem grande influência no ciclo corrente de detecção, fazendo com que cada fração de tempo realize um reconhecimento diferente sobre a imagem, em sua implementação original. Na Figura 18 é possível observar este efeito, uma vez que em um intervalo de 3 segundos objetos não são constantemente reconhecidos (ou reconhecidos erroneamente), mesmo existindo uma baixa variação na cena.

Com tais premissas em mente, a metodologia aqui proposta buscou um menor uso de processamento, possibilitando reconhecimentos mais rápidos, além de uma melhora nos resultados de detecção do *YOLO*, tanto no que se refere a falsos positivos, quanto a falsos negativos, como os vistos no exemplo da Figura 18.

5.1 ARQUITETURA PROPOSTA

De forma geral, este trabalho apresenta uma arquitetura baseada no *LIDA*, onde o modelo está em constante contato com os dados de entrada que se atualizam de forma dinâmica, semelhante ao comportamento descrito na Seção 17, ao passo que adiciona-se temporalidade de ocorrência espacial ao *YOLO*, realizando comparações com cenas préviamente inferidas.

Nesta proposta, o *YOLO* passa a ser subdividido entre os módulos da arquitetura *LIDA*, permitindo que suas funcionalidades sejam estendidas individualmente, trabalhando em para-

lelo para resolver uma mesma parte de um problema. Um exemplo é a adição de novos algoritmos em forma de *codelets* de atenção, para acompanhar regiões de interesse nas imagens, sem a utilização recorrente do *YOLO*, como em sua implementação original.

Cada um dos módulos da arquitetura possui um objetivo final diferente, porém trabalham em conjunto com o todo, por meio do ciclo de funcionamento recorrente, como demonstrado no mapeamento entre a arquitetura *LIDA* e o encéfalo humano, visto na Seção 4.

Na arquitetura proposta neste trabalho, os dados de entrada são imagens ou videos inseridas e normalizadas via módulo de memória sensorial (Figura 19, item 2), que subsequentemente envia para o memória de trabalho (ou chamado *Global Workspace*), a imagem pré-processada, como visto no fluxo apresentado na Figura 19.

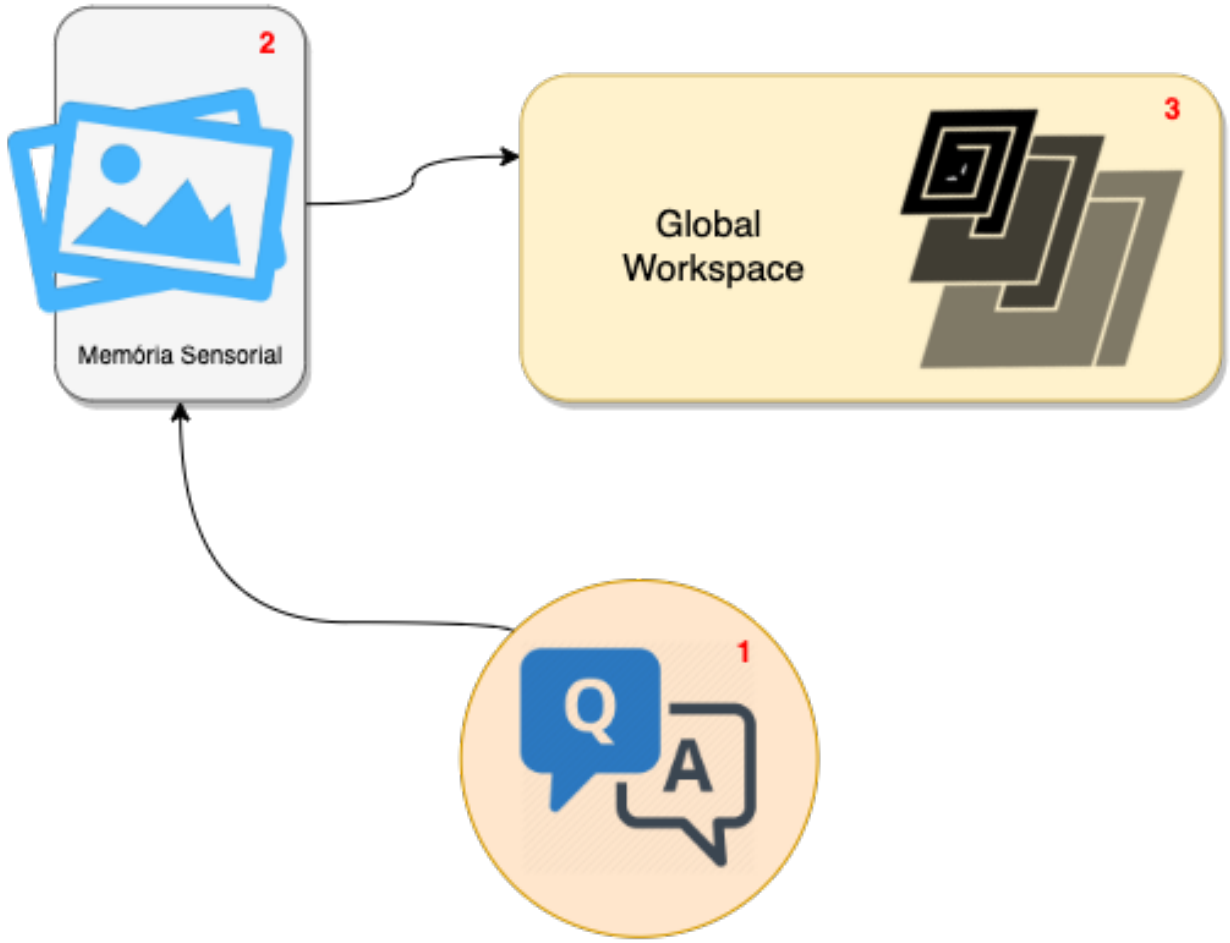


Figura 19 – Três primeiras etapas da Arquitetura proposta, baseada no modelo LIDA (FRANKLIN et al., 2016).

A memória perceptiva associativa, ou *PAM*, tem papel fundamental no processo de reconhecimento de objetos (Figura 20, módulo 5), uma vez que possui como seu *kernel* o algoritmo *YOLO*. Para isso, conta com o auxilio do módulo de memória episódica transiente, ou *T.E.M.*,

que armazena dados de cenas já processadas, afim de indicar a necessidade de uma nova execução da *PAM* (Figura 20, módulo 6 e 7).

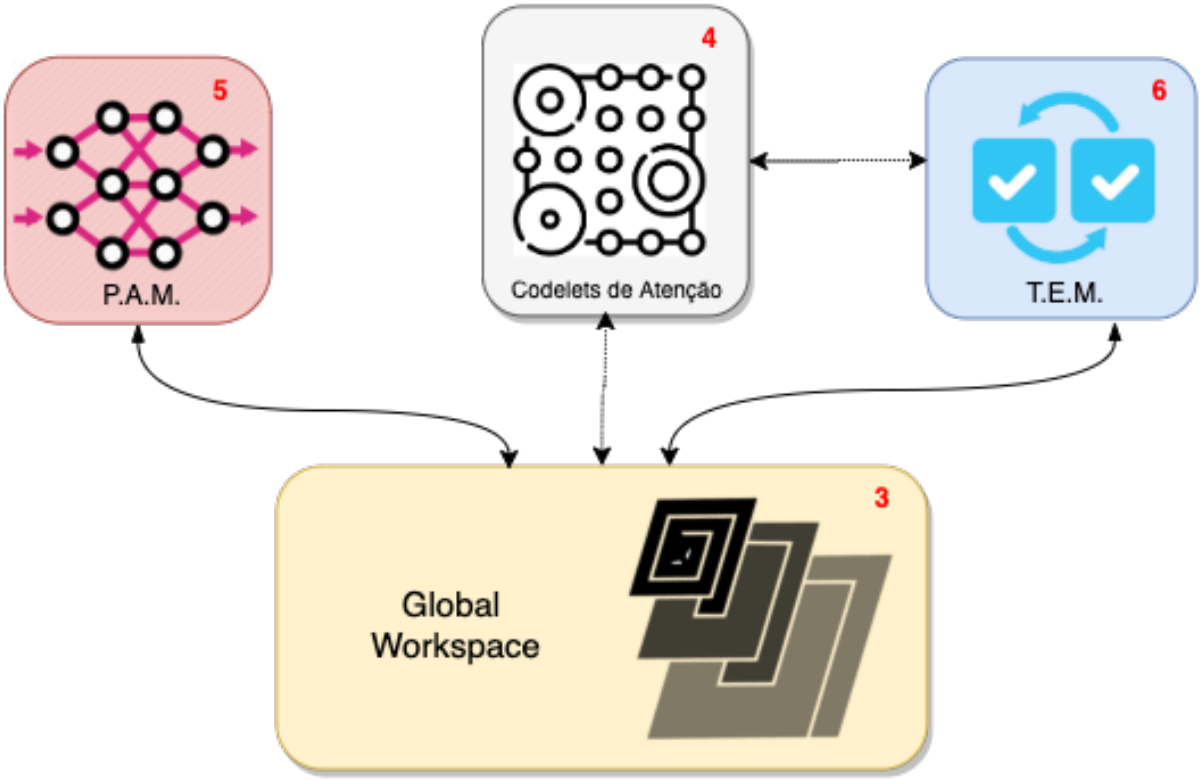


Figura 20 – Modulos principais para o reconhecimento de objetos na Arquitetura proposta, baseada no modelo LIDA (FRANKLIN et al., 2016).

Por sua vez, os dois módulos de *codlets* de atenção (o verificador de mudança de cenas e o rastreador de objetos) indicam as áreas a serem consideradas durante a fase de reconhecimento. O primeiro *codelet* de atenção se encarrega de rastrear os objetos reconhecidos pelo *YOLO* por meio de algoritmos de rastreamento, a partir de um conjunto de *bounding boxes* que delimitam as regiões reconhecidas. As probabilidades inferidas são enviadas novamente para o *Global Workspace* que encaminhará os melhores resultados calculados (maiores probabilidades) para a avaliação final. Por fim, os resultados são persistidos e apresentados via Memória declarativa, que persiste os dados e retorna para a entrada/saída padrão, como observado na Figura 21.

A arquitetura possui a mesma dinâmica de trabalho da arquitetura *LIDA*, quando todos os módulos são interligados, como apresentado na Figura 21.

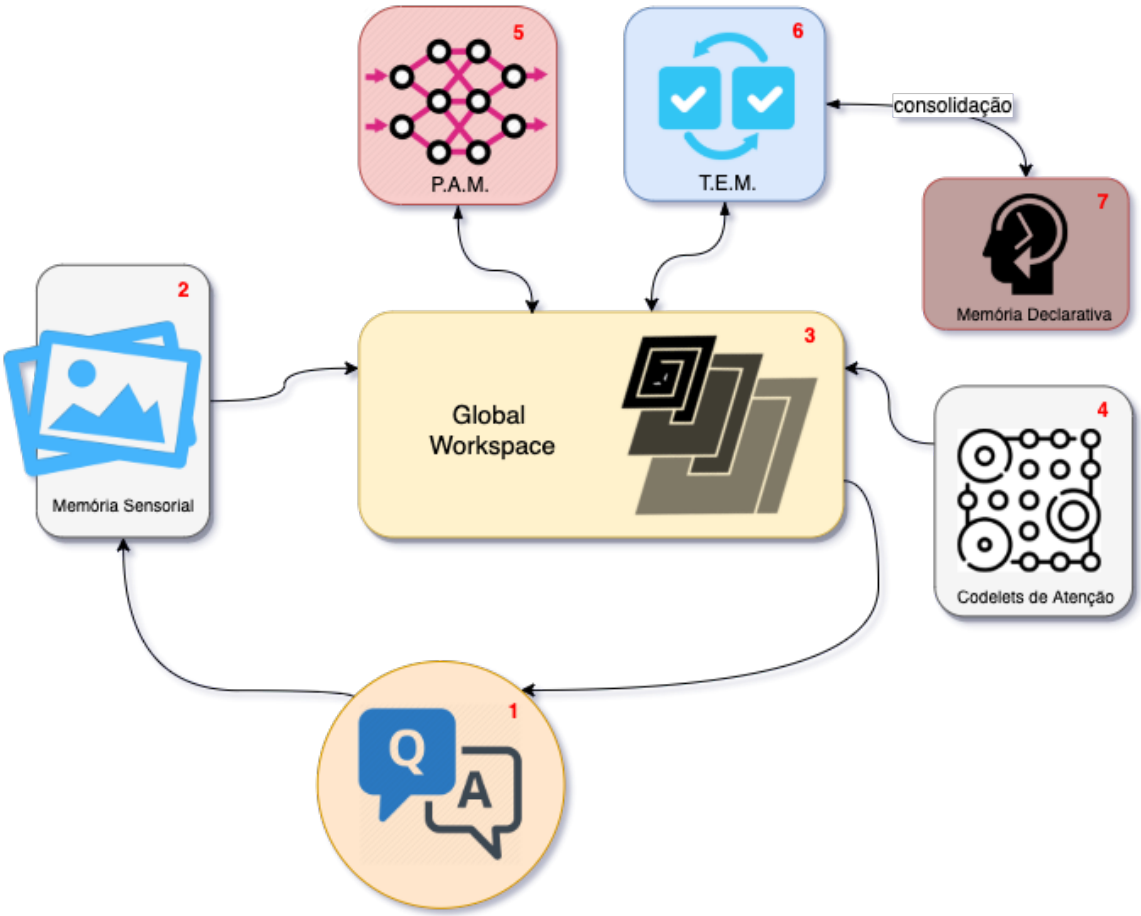


Figura 21 – Arquitetura Geral proposta, baseada no modelo LIDA (FRANKLIN et al., 2016).

Uma discussão mais profunda de cada um dos módulos é apresentada a seguir, onde as funcionalidades, algoritmos e modelos utilizados são explorados em mais detalhes.

5.1.1 Memória Sensorial

Este módulo recebe do ambiente interno e externo (Figura 21, módulo 2) imagens *RGB* para realizar o conhecimento. Essas imagens serão constamente consumidas pela memória sensorial, que realizará a leitura de forma matricial das imagens a serem enviadas para a *workspace*.

Uma vez que, o *YOLO* trabalha com imagens coloridas e em 2D, nenhuma transformação extra de domínio ou espacial será necessária, apenas um redimensionamento em seu tamanho para garantir a uniformidade dos dados de entrada, onde a escala de 240 pixels será utilizado.

5.1.2 Memória Perceptiva Associativa

A memória perceptiva associativa, uma tradução literal de *perceptual associative memory*, também conhecida pelo acrônomo *P.A.M.*, é responsável pelo módulo principal de reconhecimento visual da arquitetura (revisado na Seção 4).

Este módulo de reconhecimento, em soluções envolvendo imagens, é utilizado para realizar análise e reconhecimento de objetos de forma geral, podendo ter contato direto tanto com as imagens de entrada, quanto com informações adicionais detectadas por outros módulos, disponíveis na *Globa Workspace*.

Este agente de percepção referido necessita em seu núcleo de um algoritmo para realizar tal tarefa em tempo real, em busca de simular o comportamento que ocorre com humanos. Para isso, este trabalho propõe a utilização de um dos algoritmos mais rápidos para reconhecimento de objetos em imagens, encontrado hoje no estado da arte, que consegue ter boa assertividade em suas análises, mesmo com sua elevada velocidade que se destaca dentre seus "concorrentes".

O *YOLO*, previamente apresentado na Seção 3.2.2, em sua versão mais recente, obteve avanços importantes para seu desempenho, como o aumento da precisão para detecção de objetos pequenos atingindo níveis melhores que o algoritmo *Fast RCNN* (Capítulo 3), ou ainda, a diminuição de erros na localização de áreas de interesse a serem consideradas no processo de reconhecimento (REDMON; FARHADI, 2018).

Porém, ainda existem barreiras a serem vencidas que persistem mesmo ao longo de suas atualizações, como a dificuldade em detectar continuamente todos os objetos de interesse na cena, ou a necessidade de um hardware específico, ainda pouco acessível principalmente em dispositivos móveis, para desempenhar os processos com fluidez (REDMON; FARHADI, 2018). Tais problemas, podem ter relação não somente com ajustes realizados na quantidade de camadas de sua rede convolucional, nem na quantidade de níveis visitados para calcular suas *bounding boxes*, abordagens atualmente aplicadas por seus autores. Olhando por uma outra via, este trabalho propõe uma nova arquitetura para o *YOLO* com sua estrutura de reconhecimento de objetos integrando a *P.A.M.* da arquitetura vista na Figura (21, módulo 5), em busca de uma melhora tanto na continuidade de reconhecimento dos objetos, quanto em seu tempo de processamento.

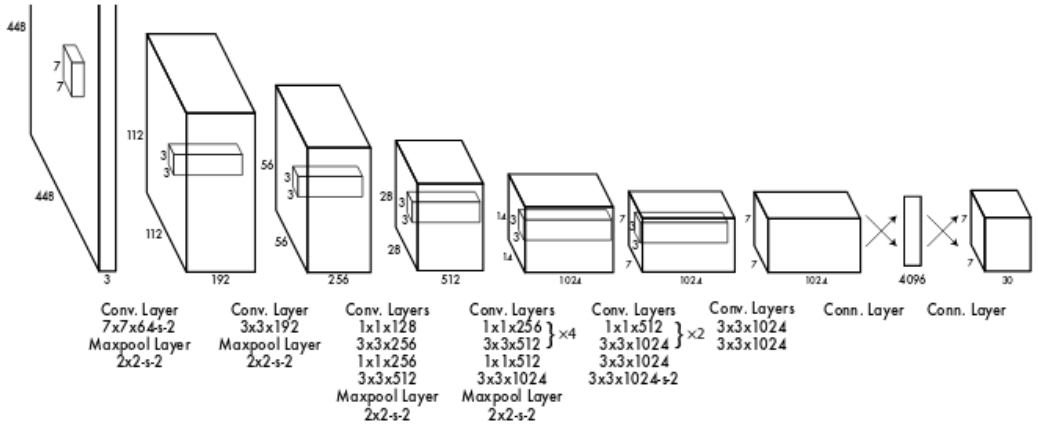


Figura 22 – Arquitetura original do *YOLO*, adaptado de (REDMON et al., 2016).

Para isso, foi validada a seguinte abordagem. Manter as características originais da rede, vista na Figura (22), guardando os vetores de delimitação de objetos conhecidos e aplicando uma perturbação sobre cada um, buscando reutilizá-los em *frames* futuros indicados pelos *codelets* de atenção por meio do uso de algoritmos de rastreamento de objetos (Seção 2.1.4), quando não houverem mudanças significativas na cena. Esta abordagem parte do princípio de que, potencialmente, o mesmo objeto ainda deve existir em *frames* subsequentes na mesma localização encontrado anteriormente, ou em uma região próxima, quando não ocorrerem alterações significativas na continuidade da cena. Com este procedimento, espera-se que ocorra uma diminuição nos casos de não continuidade no reconhecimento dos objetos, como representado na Figura (18), além de uma redução significativa no seu tempo de processamento, uma vez que um modelo mais simples será aplicado na localização dos objetos em cenas contínuas para seguir os objetos, não se fazendo necessário recalcular todas as estimativas de objetos existentes a todo momento.

A *P.A.M* recebe *Grids* de células, onde cada uma é um *bounding box*. Cada célula recebida possui 5 valores $(x,\ y,\ w,\ h) \in R$ e o valor de confiança daquele *Grid* contendo um objeto de interesse. As coordenadas *(x,y)* representam o centro da demarcação de onde está o objeto de interesse e *(w,h)* representam a largura e altura desse objeto, respectivamente. Por sua vez, o valor de confiança da célula contendo um objeto de interesse é comparado a partir do IoU (revisado na Seção 2.1), entre a área de demarcação e o padrão ouro, realizada pelos *Codelets* de Atenção.

Por fim, o resultado obtido é salvo na memória episódia transiente, retornando o controle do processo para a mem'ria de trabalho.

5.1.3 Memória Episódica Transiente

Como definido na Seção (4), a Memória Episódica armazena eventos representados por processos temporais ocorridos em um período recente. A partir disso, é possível adicionar ao *YOLO* um novo comportamento, que leva em consideração objetos detectados em frames anteriores, adicionando pequenas perturbações nos delimitadores, quando a cena não sofrer grandes alterações.

Este módulo foi introduzido em busca de melhorar a identificação de objetos que já ocorreram na mesma cena, uma vez que por conta da característica que o *YOLO* possui, na qual *olha para todo frame uma vez*, as detecções sofrem variações mesmo sem grandes mudanças na cena, como visto na Figura 18, pois em cada fração de tempo um novo reconhecimento é realizado sem considerar o histórico recente.

Essas informações armazenadas são utilizadas pelos codelets de atenção, mais especificamente pelo responsável no rastreamento de objetos préviamente reconhecidos, enquando não ocorrem grandes mudanças na cena.

Figura 23 – Exemplo de transição que não indica mudanças de cena, quando o *codelet* de atenção é utilizado

A Figura 23 apresenta uma sequência de imagens que com o suporte dos *codelets* de atenção, têm seus objetos rastreados com a utilização dos dados armazenados na memória episódica transiente.

5.1.4 Codelets de Atenção

Os *Codelets* de Atenção são responsáveis por indicar áreas de interesse na imagem de entrada. Esse processo é realizado para que o espaço a ser analisado seja restrito e assertivo,

aumentando assim a velocidade de processamento e reduzindo possíveis erros, além de permitir a utilização dos dados armazenados na memória recente (*TEM*).

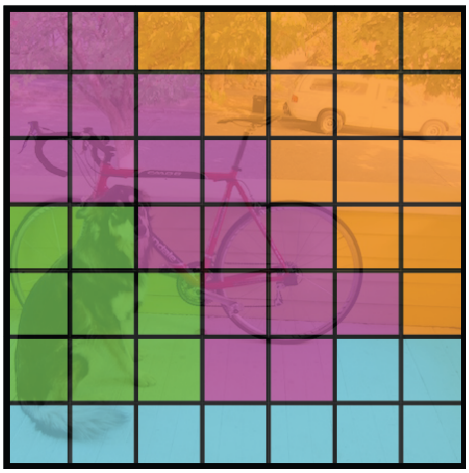


Figura 24 – Representação do mapa de probabilidades gerado pelo *YOLO*

Para este módulo, as duas abordagens indicadas nas Seção (5.1.2) com a utilização de algoritmos apresentados no Capítulo 2 foram tomadas.

O primeiro *codelet* de atenção, aqui chamado de *change detector*, é responsável por comparar os frames dos videos de entrada, verificando se ocorreram mudanças consideráveis buscando identificar quando novos objetos apareceram na cena e antigos deixaram de existir.

Diferentes métodos para cálculo de diferença entre imagens foram utilizados como base para os experimentos, com o objetivo de identificar aqueles de melhor desempenho (BRADSKI; KAEHLER, 2008). Uma vez que tais métodos não poderiam ter altos custos computacionais, foram selecionados apenas modelos de comparação de histogramas, sendo eles os modelos de Correlação, Chi-quadrado, interseção e a distância de Bhattacharyya, apresentados na Seção 2.1.3.

Neste trabalho, inferências de *bounding boxes* previamente calculadas em cena são utilizadas com o objetivo de introduzir o conceito de memória recente para reconhecimento de objetos. Objetos identificados pelo algoritmo presente na *PAM* são mantidas em memória recente (*TEM*) para que nas imagens subsequêntes de uma cena sejam utilizados novamente com perturbações, buscando encontrar o mesmo objeto em outros *frames*.

Para isso, um outro *codelet* de atenção também está presente no modelo proposto responsável por rastrear objetos préviamente detectados no video em questão. Neste caso, os algoritmos apresentados na Seção 2.1.4 foram utilizados, com o objetivo de verificar o desempenho de cada um na arquitetura proposta, sendo eles o BOOSTING, MIL, KCF, TLD, MEDIANFLOW e CSRT.

O mapa de probabilidades (Figura 24), contendo os objetos previamente reconhecidos gerados na *PAM* e armazenadas na *TEM*, são utilizados pelos algoritmos aqui implementados para rastrear objetos em próximos frames, sem a utilização da *PAM*.

5.1.5 Global Workspace

Na arquitetura proposta, a *Global Workspace* (Figura 21, módulo 3) mantém dados, *features* e probabilidades no ciclo de reconhecimento, sendo dito como consciente, formando o centro do Modelo Situacional Atual. Tais informações, ficam disponíveis para uso das memórias periféricas até que uma nova chamada a memória *P.A.M.* se faça necessária, indicada pelos *codeletes* de atenção.

Enquanto as memórias Perceptiva e Episódica processam e armazenam as probabilidades do objeto pertencer a uma determinada classe e se a cena atual sofreu grandes alterações nos últimos *frames* (Figura 21, módulos 5 e 6), a *Global Workspace* realiza a ligação entre estes módulos (Figura 21, módulo 3), além de receber os tensores calculados pelos *Codelets* de atenção (Figura 21, módulo 4) que também serão utilizados para o rastreamento dos objetos.

Após o processamento dos *Grids* de células pelos *Codelets* de atenção, a *Global Workspace* receberá os tensores, Expressão (3), e os encaminhará para a Memória Perceptiva Associativa para os cálculos de probabilidades sobre as *bounding boxes* pré-selecionadas.

Em paralelo, a *Global Workspace* recebe da Memória Episódica Transiente o tensor de memória recente com as probabilidades calculadas sobre os *frames* anteriores, para serem encaminhados para o *codelet* de atenção responsável pelo rastreamento dos objetos já reconhecidos. A *Global Workspace* também é responsável por avaliar se o retorno do *codelet* de atenção que verifica mudança de cena, indicou tal ocorrência.

Após a avaliação de todos os *Grids* da imagem, um vetor de tensores da memória episódica transiente, armazena todos os valores de probabilidades calculados. Uma ordenação realizada sobre as probabilidades é realizada sobre os vetores, indicando os objetos reconhecidos na cena em questão a partir de um limiar definido. Ao final do processo, a *Global Workspace* enviará avaliará se ciclo atual é o utilmo da execução, ou se ainda existem imagens a serem processada.

Caso nenhuma outra imagem ainda esteja em fila de processamento, os vetores da memória *TEM* são armazenados no disco local com os resultados calculados para todos os *frames* processados.

6 PROPOSTA EXPERIMENTAL

Em busca de evoluir com ferramentas de reconhecimento de objetos, um novo modelo é proposto neste trabalho utilizando os avanços obtidos pela neurociência e computação cognitiva. Para isso, a arquitetura *LIDA* (Seção 4) será utilizada como base para uma nova implementação do algoitmo de reconhecimento de objetos *YOLO* (Seção 3.2.2).

A arquitetura proposta por este trabalho (Figura 25), no geral, busca um desempenho equivalente, ou minimamente superior, em detecção de objetos, reduzindo problemas de não continuidade de reconhecimento de um mesmo objeto, em relação a implementação original do *YOLO* em sua 3ª versão. Além disso, um aumento na velocidade de processamento é esperado, uma vez que o *YOLO* não deverá mais ser executado a cada *frame* de video, considerando que os *codelets* de atenção indiquem a não necessidade desta nova execução.

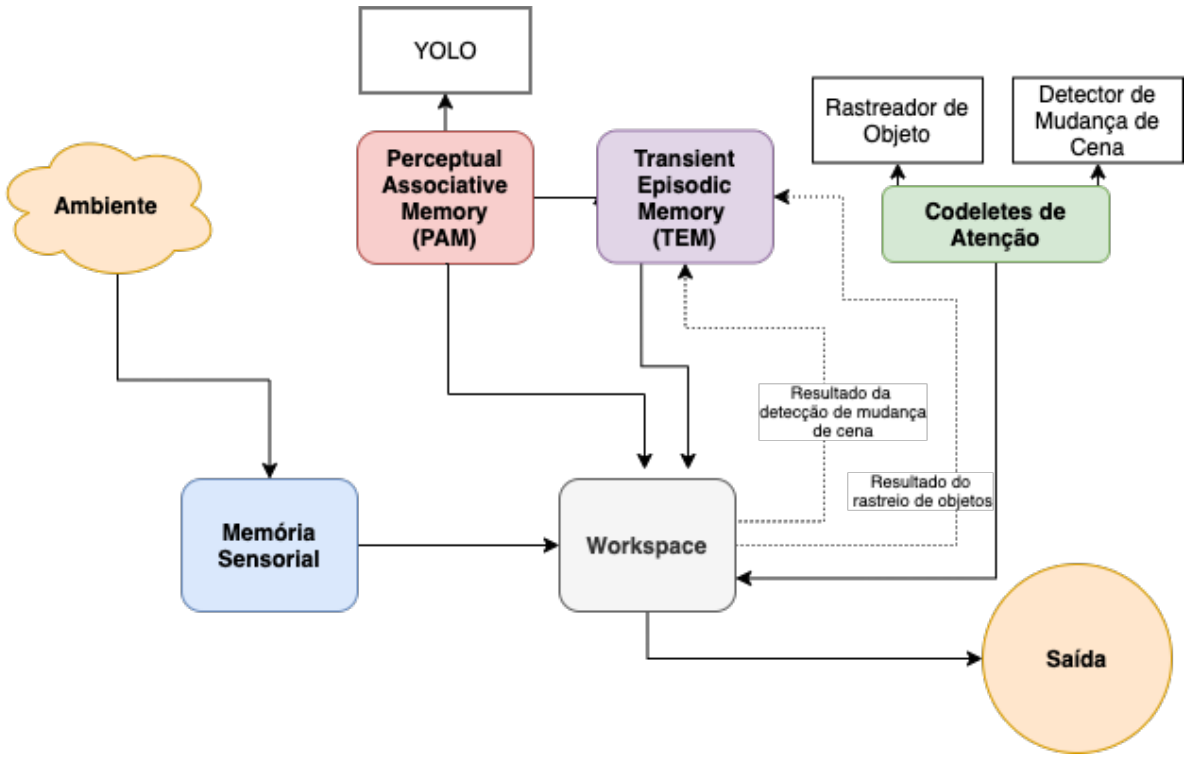


Figura 25 – Arquitetura Proposta Final

Para validar esta hipóteses, os experimentos foram programados para serem realizados sobre a base de videos *TV77 - Videos Object Dataset* (vista na Seção 6.0.1). Primeiramente, para cada diretório da base, o *YOLO* foi executado em sua versão sem alterações para obter seu desempenho sem alterações. Em seguida, para o modelo proposto, visto na Figura 25, foi executado variando o núcleo dos *codelets* de atenção. Para cada video, o modelo foi executado com

todos os algoritmos apresentados para o ratreamento de objetos (*CSRT, KCF, Boosting, MIL, TLD, MedianFlow, MOSSE*), onde para cada um deles, todos os métodos de detecção de mudança de cena foram experimentados (*Correlation, Chi-Square, Intersection, Bhattacharyya*). Assim, no Capítulo 8 é possível analisar os diferentes comportamentos, afim de indicar as melhores combinações encontradas para o uso da arquitetura aqui proposta.

Atualmente, o *YOLO* atinge um desempenho superior ao de outros algoritmos de reconhecimento de objetos, porém ainda é considerado o uso de hardware específico e ainda inacessível para a maioria das aplicações, como aquelas embarcadas em dispositivos portáteis. Há expectativa de que o modelo aqui proposto possa nortear um novo caminho a ser seguido para solucionar este problema, possibilitando o uso do *YOLO*, ou outras redes neurais, em suas modelagens completas, não mais utilizando versões reduzidas e menos assertivas, como ocorre atualmente.

6.0.1 TV77: Video Object Tracking Dataset

A *TV77* faz parte de um projeto de pesquisa que objetiva construir uma base de imagens anotada e classificada a partir de outras já existentes para auxiliar na organização e disponibilização de um alto volume de imagens para pesquisadores no desenvolvimento de suas soluções (MADER, 2018). As imagens e suas anotações são a base para um desafios de rastreamento de objetos realizados na plataforma *Kaggle* (ANALIDE; KIM, 2017). Um exemplo da base pode ser visto na Figura 26.

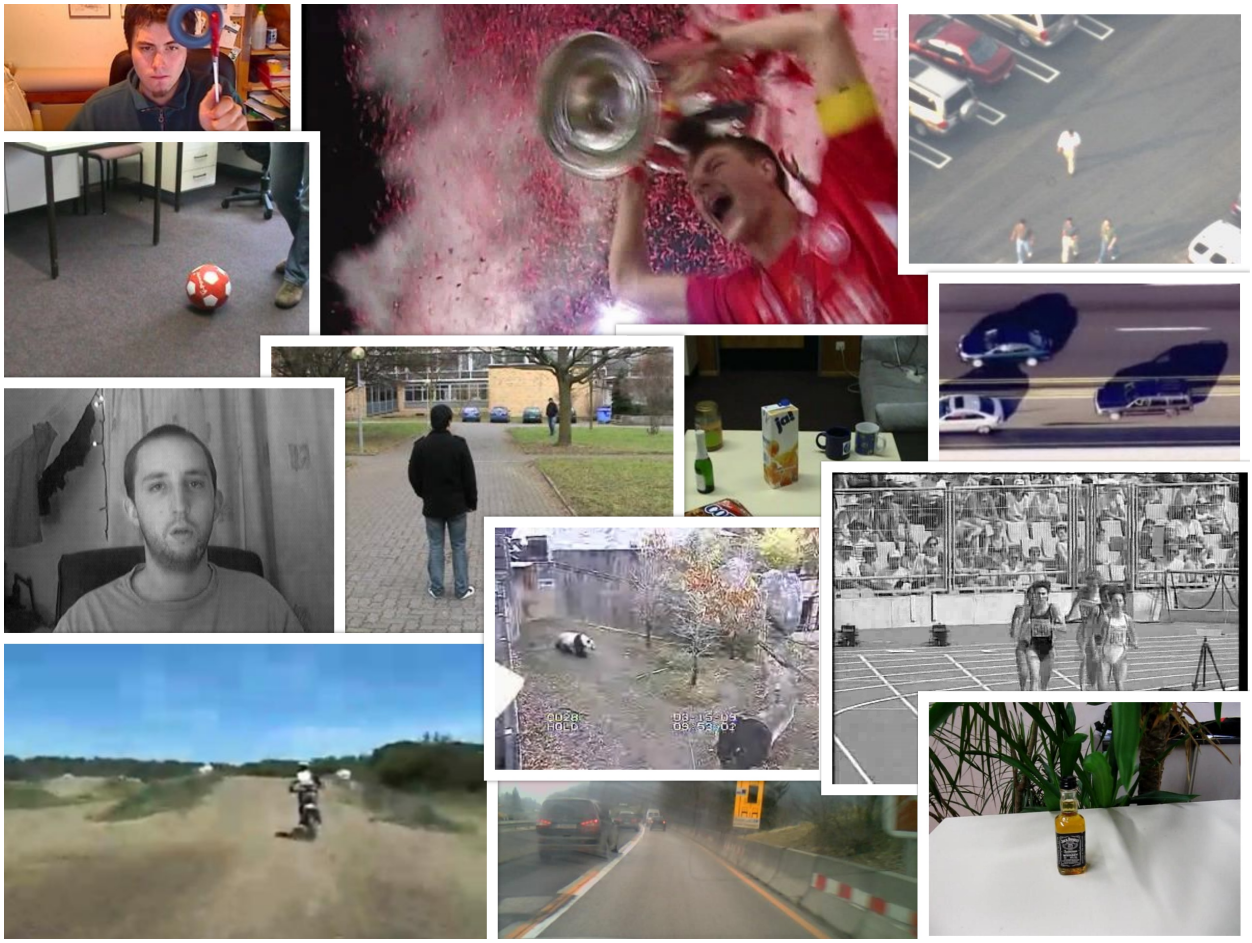


Figura 26 – Exemplo dos videos contidos na Base de Imagens *TV77*

7 RESULTADOS EXPERIMENTAIS

Neste Capítulo são apresentados os resultados obtidos por meio da execução dos experimentos realizados com o modelo proposto na Seção 15.

Para fins de comparação, diferentes algoritmos foram utilizados como base dos codelets de atenção, tanto para o rastreamento de objetos, quanto para a validação de mudança de cena sobre a mesma base de imagens, a *TV77* vista na Seção 26.

Dessa forma, os experimentos foram subdivididos entre os diferentes codelets de atenção de rastreamento de objetos, sendo eles, o BOOSTING, MIL, KCF, TLD, MEDIANFLOW e CSRT, apresentados nas Seções 2.1.4 e 6. Para todos eles foi utilizado o *YOLO v3* como *kernel* da memória associativa perceptual (PAM), utilizando todos os objetos detectados pelo mesmo com nível de certeza acima de 70%. Este valor foi assumido para padronizar e equalizar as condições dos experimentos.

Por sua vez, para cada rastreador de objetos, foi utilizado um dos seguintes *codelets* de atenção de validação de mudança de cena, sendo eles, o método de Correlação, Chi-Square, Interseção, e a distância de Bhattacharyya. Neste caso, limiares foram assumidos empiricamente para cada um dos métodos, de a forma a permitir uma padronização nos resultados gerados, sendo eles respectivamente $(0.97, 5, 19,$ e $0.16)$.

Além dos algoritmos testados, o *YOLO* foi executado sem alterações em sua ultima versão estável (versão 3), sobre o mesmo *dataset* TV77, sobre as mesmas condições das demais execuções. Além disso, todos os frames forma normalizados para 240 pixels em todas as execuções, valor comumente utilizado nos experimentos em geral (Seção 3).

Por fim, o hardware utilizado em todos os testes é apresentada na Tabela 2.

Tabela 2 – Configuração do hardware utilizado nos experimentos

Processador	2,7 GHz Dual-Core Intel Core i5
Memória	8 GB 1867 MHz DDR3/SSD MAC
Placa Gráfica	Intel Iris Graphics 6100 1536 MB
Software	Python 3.7.3/OpenCV 3.4.2/PyCharm 2019.3
Sistema Operacional	MacOS Catalina 10.15.2

Fonte: Autor

7.1 EXPERIMENTO YOLO

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* em sua versão original sem modificações e sem a utilização da arquitetura *LIDA*.

Na Figura 27 é possível observar o tempo de execução, em segundos, do *YOLO* para cada um dos *frames*.

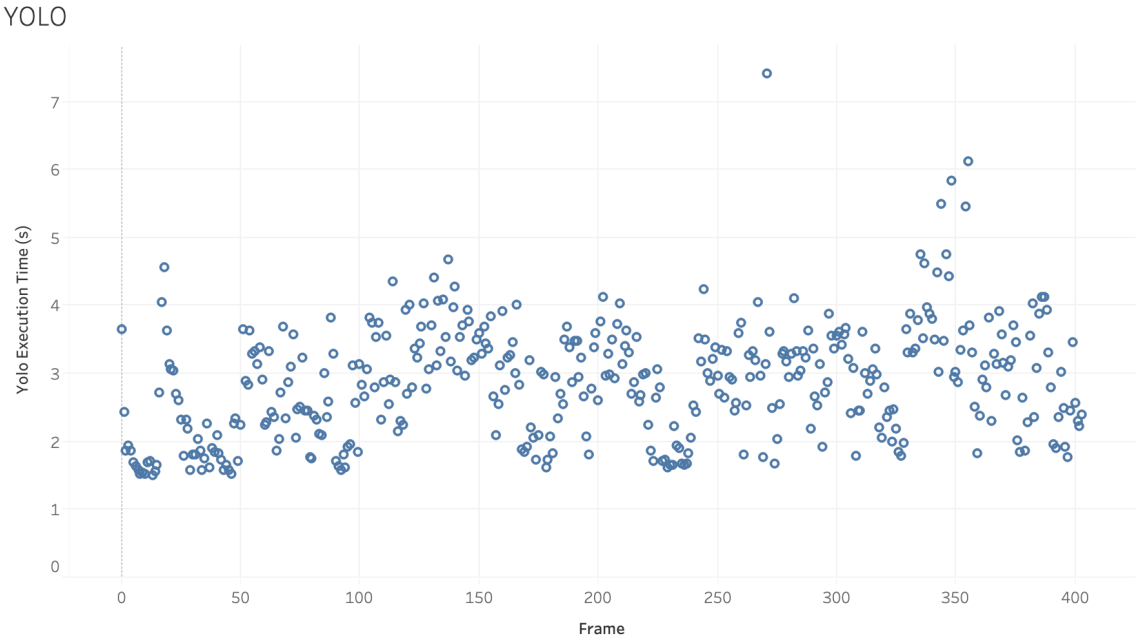


Figura 27 – Resultados utilizando apenas o YOLO

O desempenho do *YOLO* mensurado a partir do IoU com o padrão-ouro da base de imagens, pode ser observado na Figura 28.

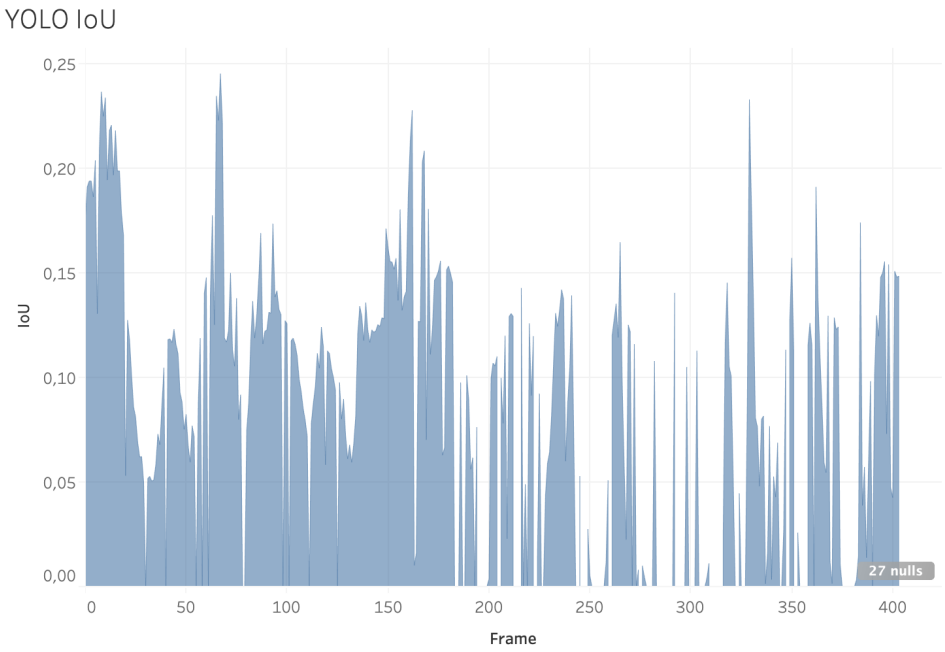


Figura 28 – Assertividade mensurada com IoU, utilizando apenas o YOLO

A Figura 29 mostra o tempo total em segundos, gasto pelo *YOLO* para ser executado sobre todos os *frames*, sendo $1171s$, ou $19{,}5$ minutos.

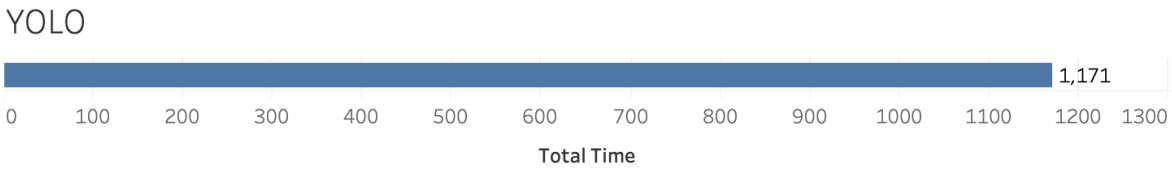


Figura 29 – Tempo total gasto pelo YOLO para a execução em todos os *frames*

7.2 EXPERIMENTO BOOSTING

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *Boosting* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *Boosting* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.2.1 BOOSTING + Correlation

Na Figura 30, no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

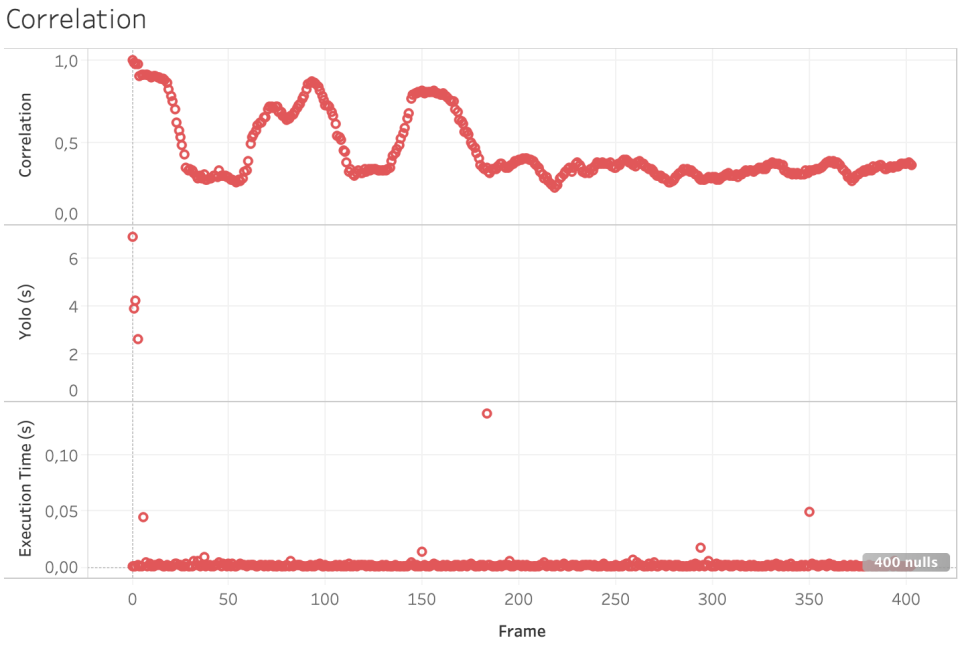


Figura 30 – Resultados utilizando o BOOSTING e Correlation

7.2.2 BOOSTING + Chi-Square

Na Figura 31 é possível observar no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

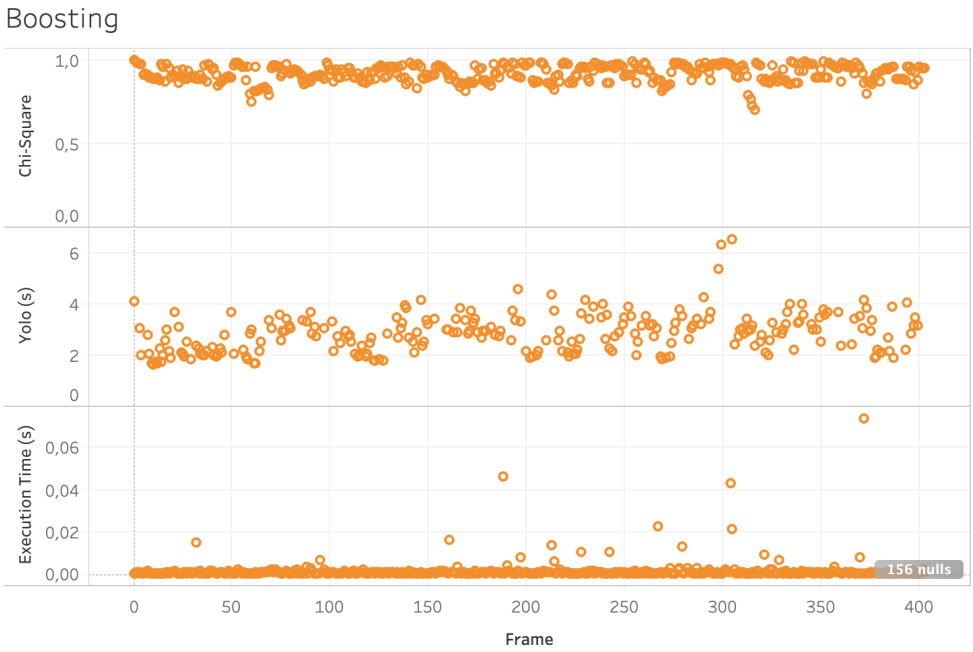


Figura 31 – Resultados utilizando o BOOSTING e Chi-Square

7.2.3 BOOSTING + Intersection

Na Figura 32 no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

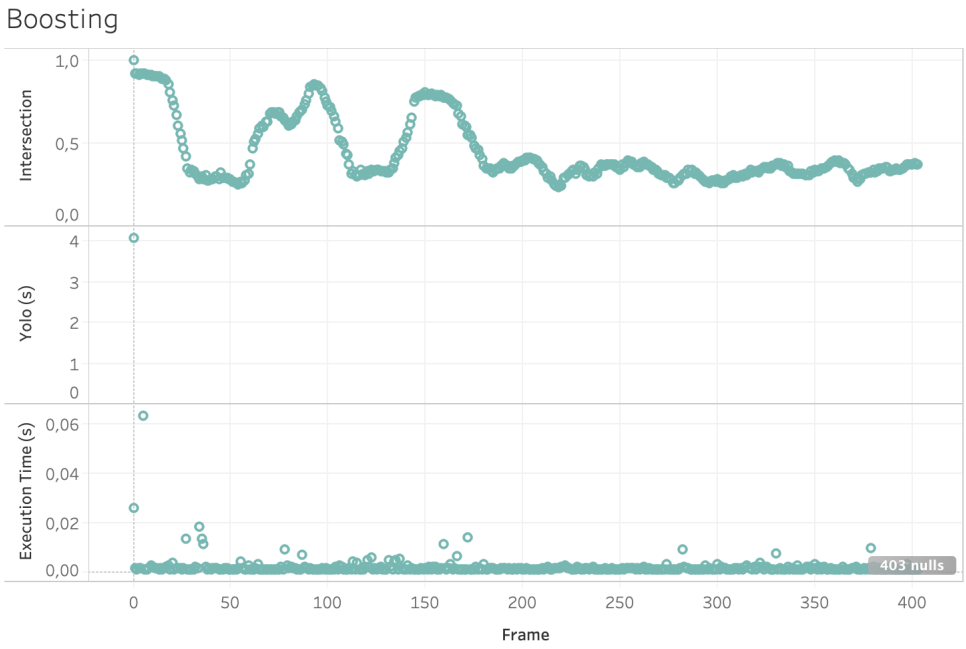


Figura 32 – Resultados utilizando o BOOSTING e Intersection

7.2.4 BOOSTING + Bhattacharyya distance

Na Figura 33 no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

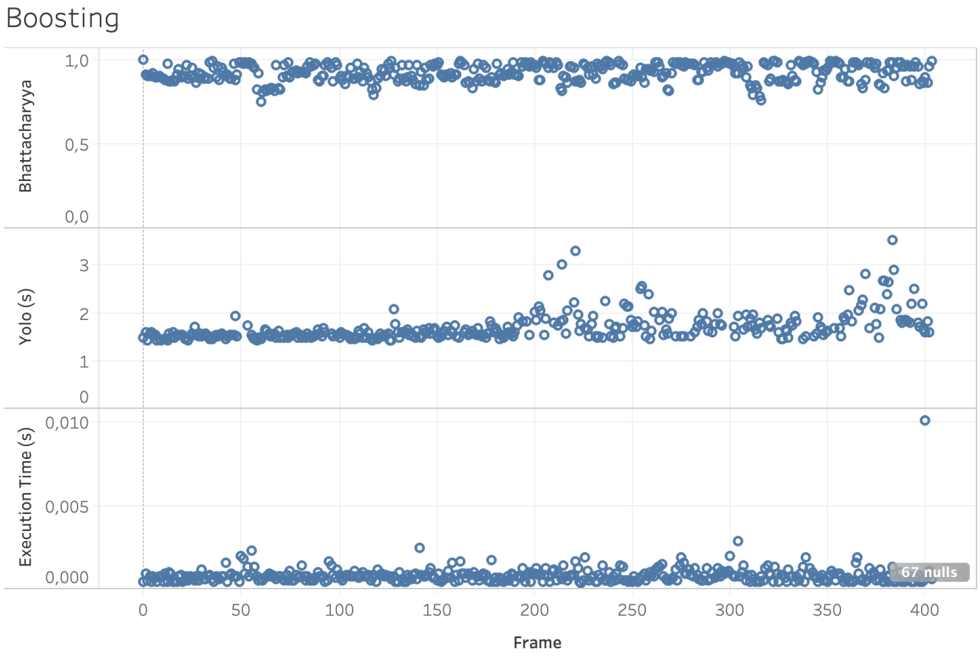


Figura 33 – Resultados utilizando o BOOSTING e Bhattacharyya

7.3 EXPERIMENTO MIL

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *MIL (Multiple Instance Learning)* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *MIL* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.3.1 MIL + Correlation

Na Figura 34, no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

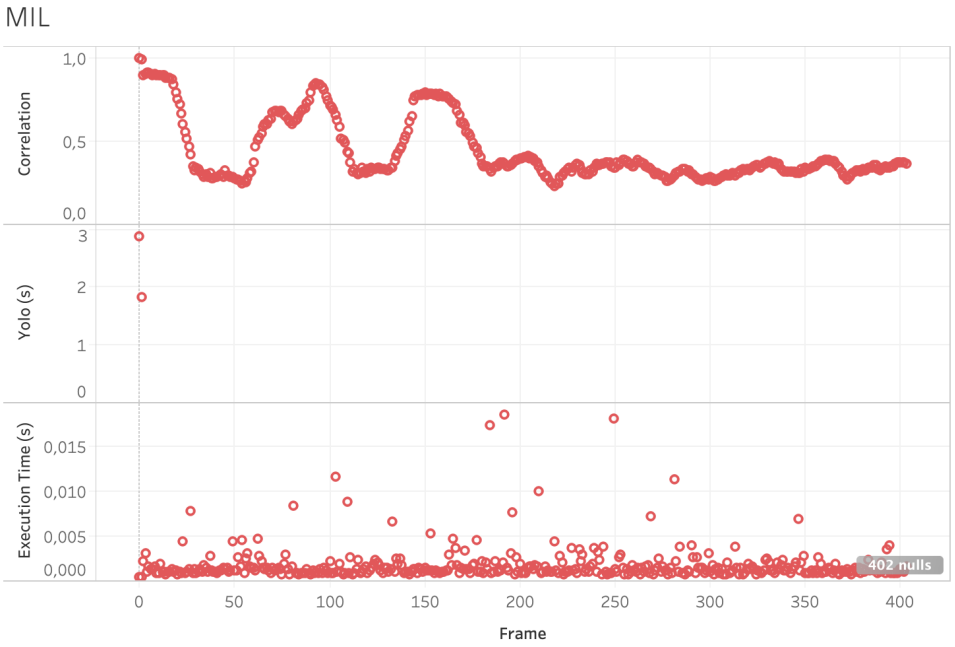


Figura 34 – Resultados utilizando o MIL e Correlation

7.3.2 MIL + Chi-Square

Na Figura 35 é possível observar no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

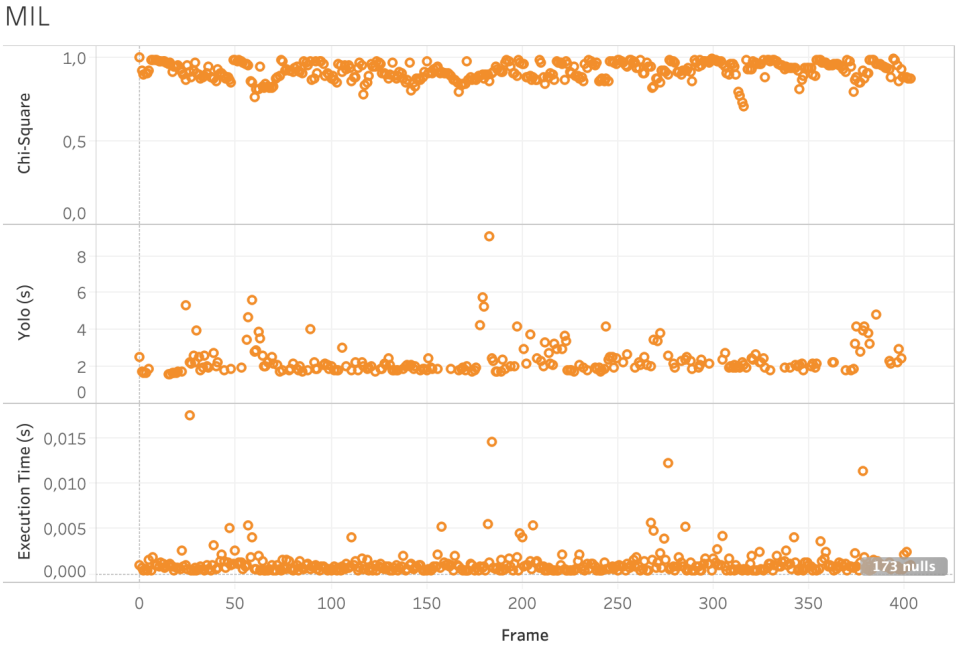


Figura 35 – Resultados utilizando o MIL e Chi-Square

7.3.3 MIL + Intersection

Na Figura 36 no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

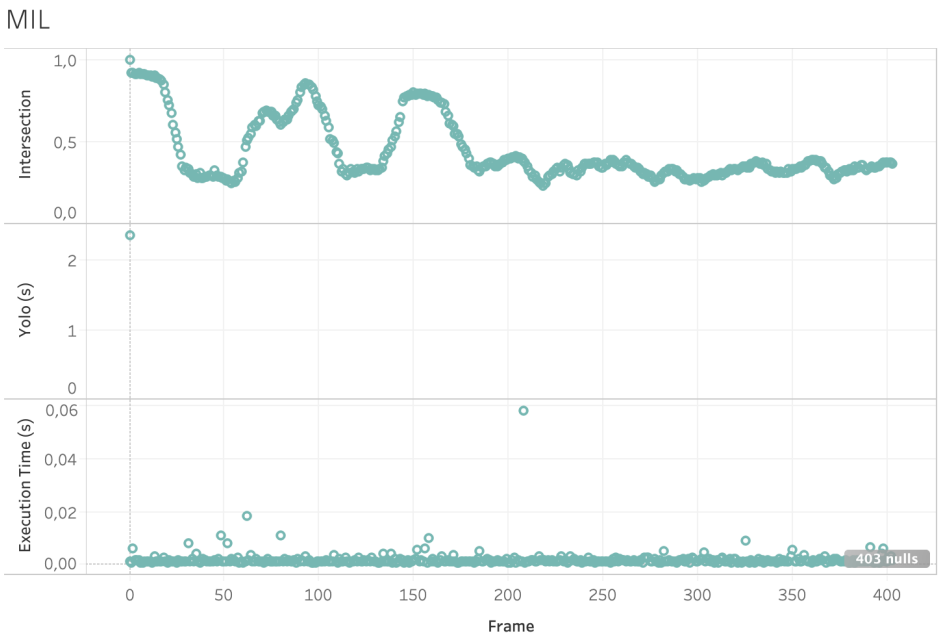


Figura 36 – Resultados utilizando o MIL e Intersection

7.3.4 MIL + Bhattacharyya distance

Na Figura 37 no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

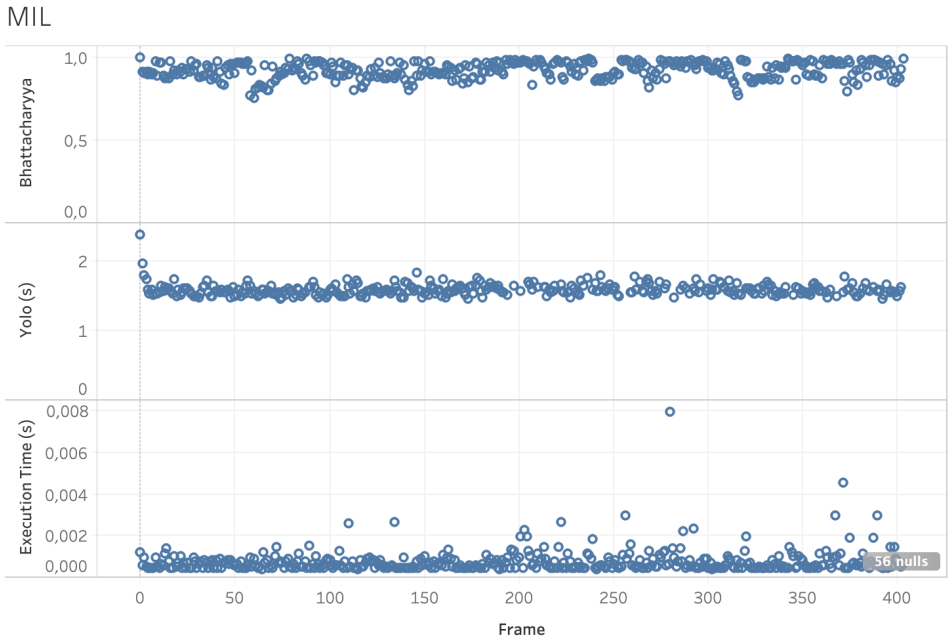


Figura 37 – Resultados utilizando o MIL e Bhattacharyya

7.4 EXPERIMENTO KCF

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *KCF (Kernelized Correlation Filters)* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *KCF* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.4.1 KCF + Correlation

Na Figura 38 no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

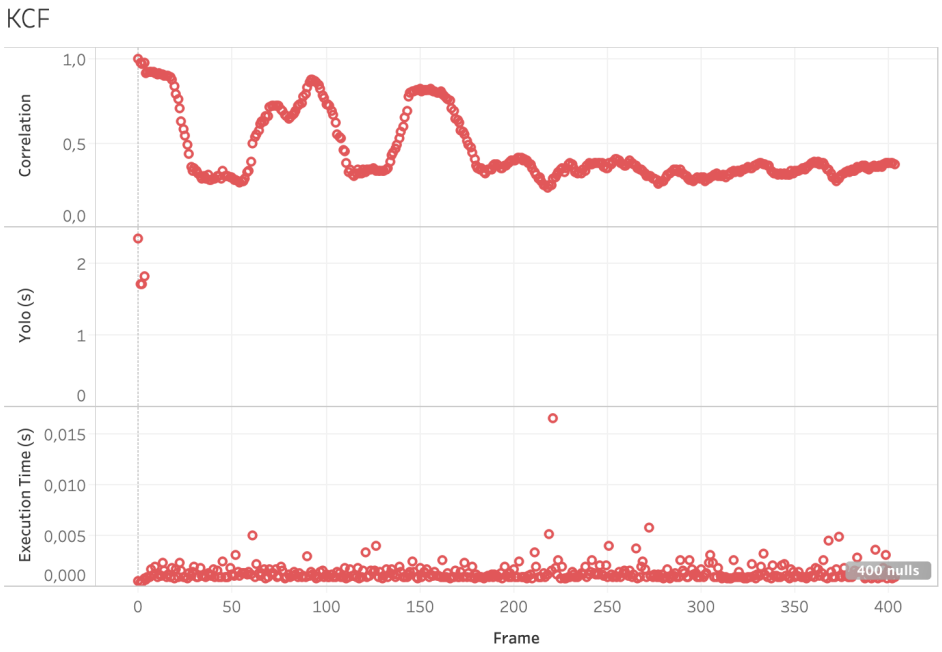


Figura 38 – Resultados utilizando o KCF e Correlation

7.4.2 KCF + Chi-Square

Na Figura 39 no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

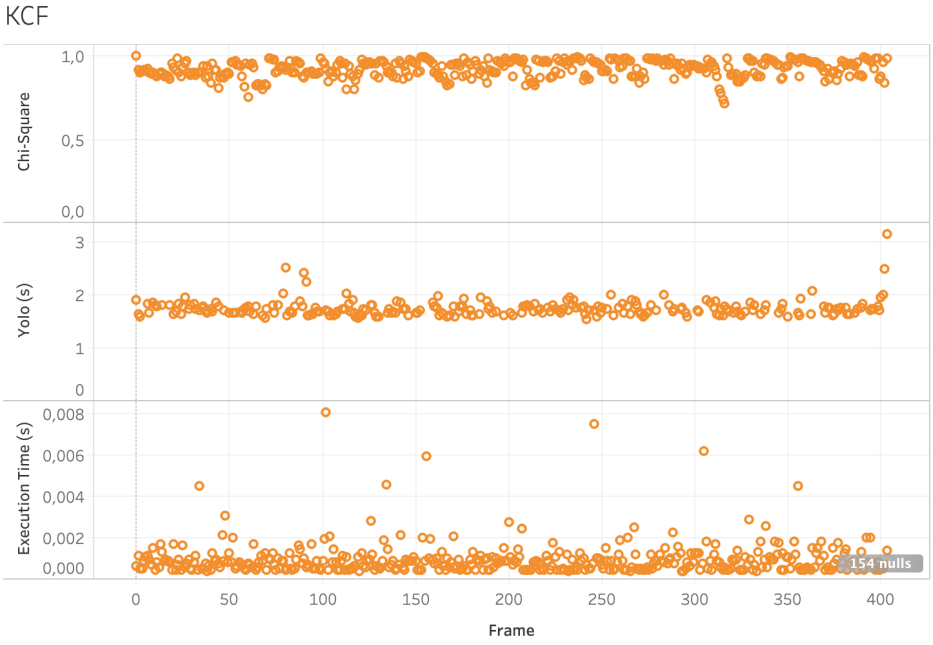


Figura 39 – Resultados utilizando o KCF e Chi-Square

7.4.3 KCF + Intersection

Na Figura 40 no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

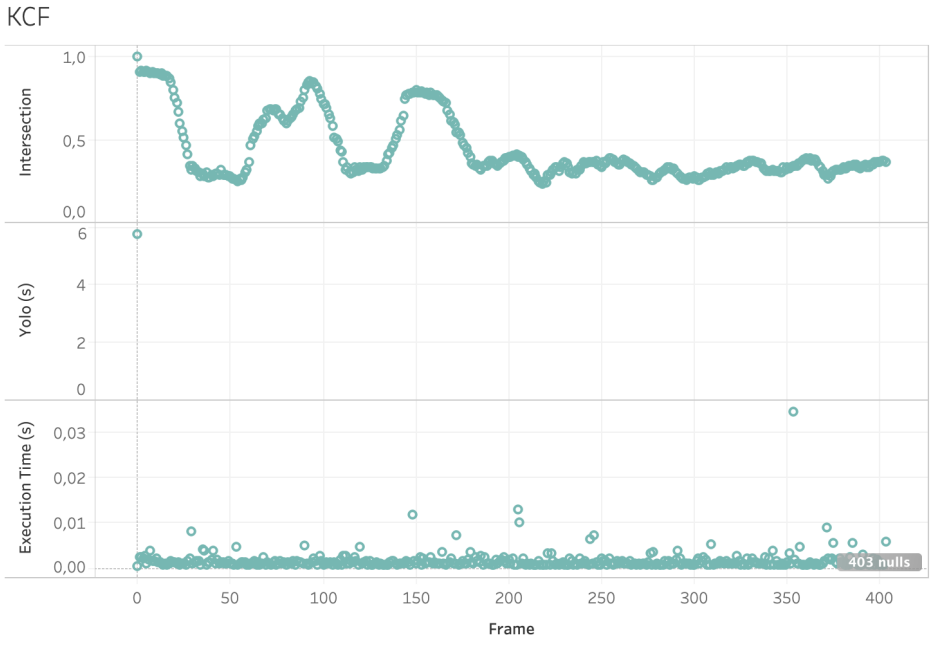


Figura 40 – Resultados utilizando o KCF e Intersection

7.4.4 KCF + Bhattacharyya distance

Na Figura 41 no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

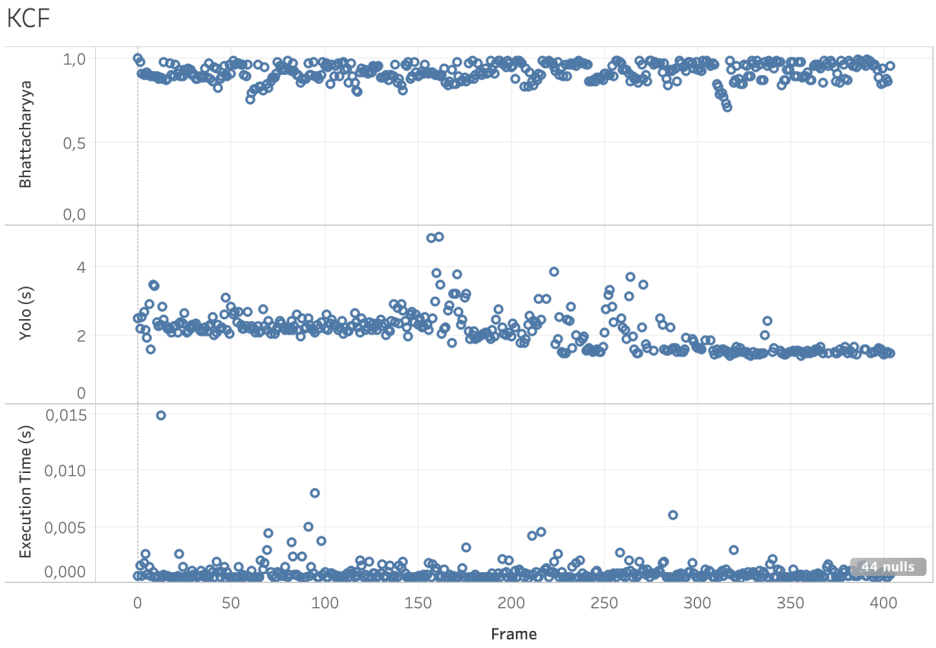


Figura 41 – Resultados utilizando o KCF e Bhattacharyya

7.5 EXPERIMENTO TLD

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *TLD (Tracking, learning and detection)* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *TLD* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.5.1 TLD + Correlation

Na Figura 42, é possível observar no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

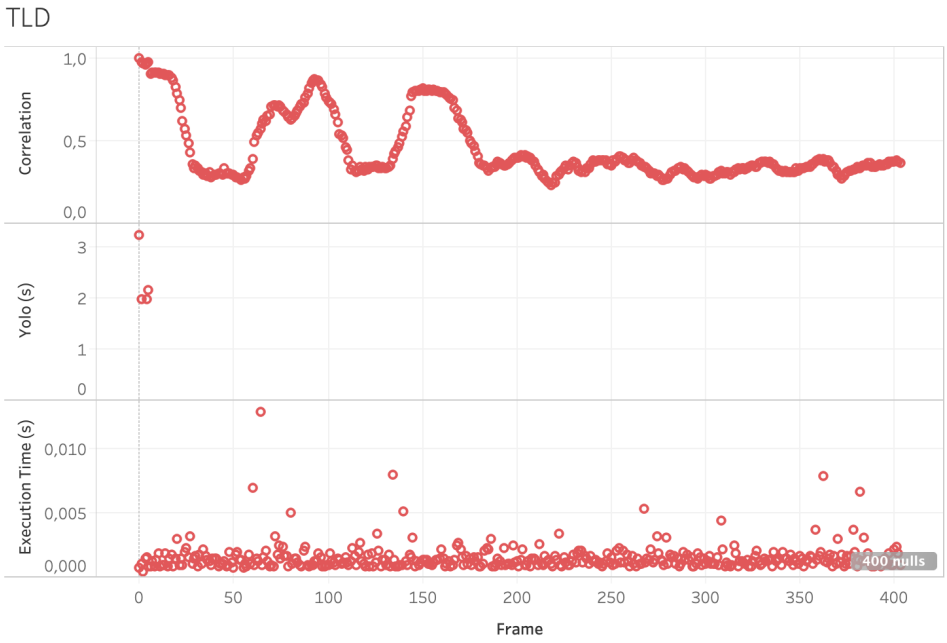


Figura 42 – Resultados utilizando o TLD e Correlation

7.5.2 TLD + Chi-Square

Na Figura 43 no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

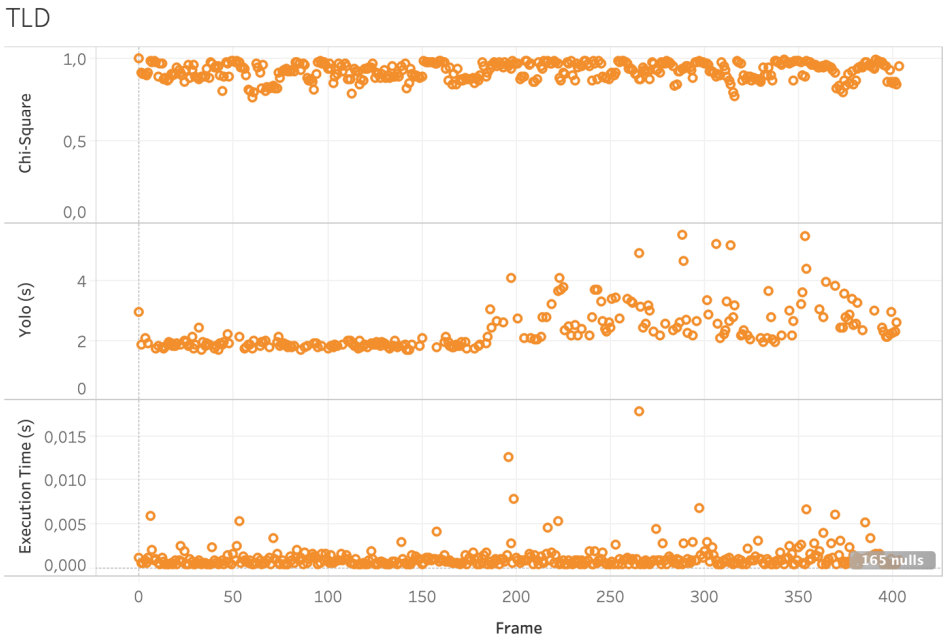


Figura 43 – Resultados utilizando o TLD e Chi-Square

7.5.3 TLD + Intersection

Na Figura 44 no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

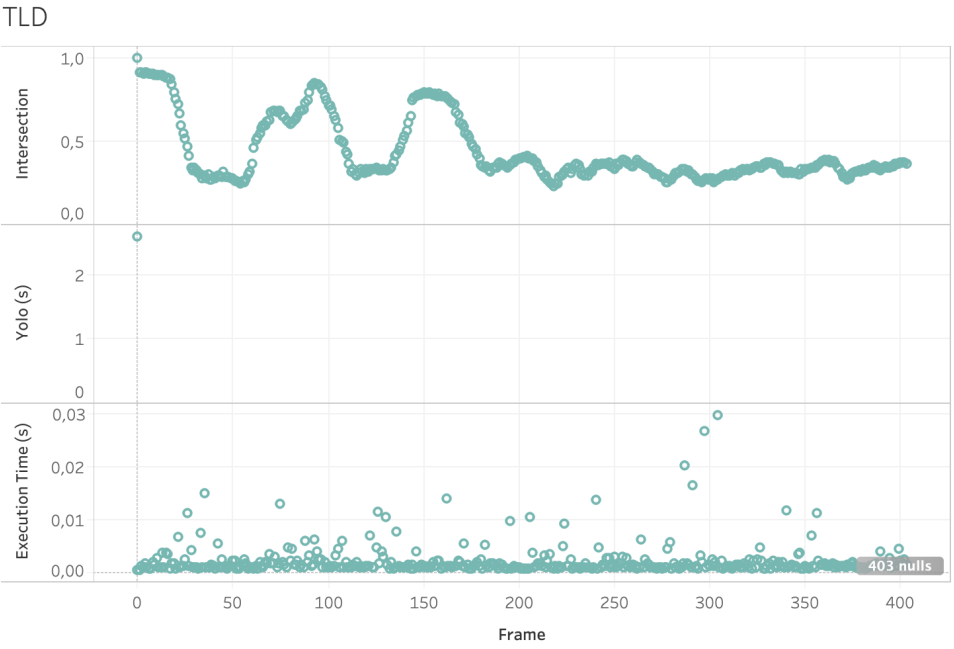


Figura 44 – Resultados utilizando o TLD e Intersection

7.5.4 TLD + Bhattacharyya distance

Na Figura 45, é possível observar no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

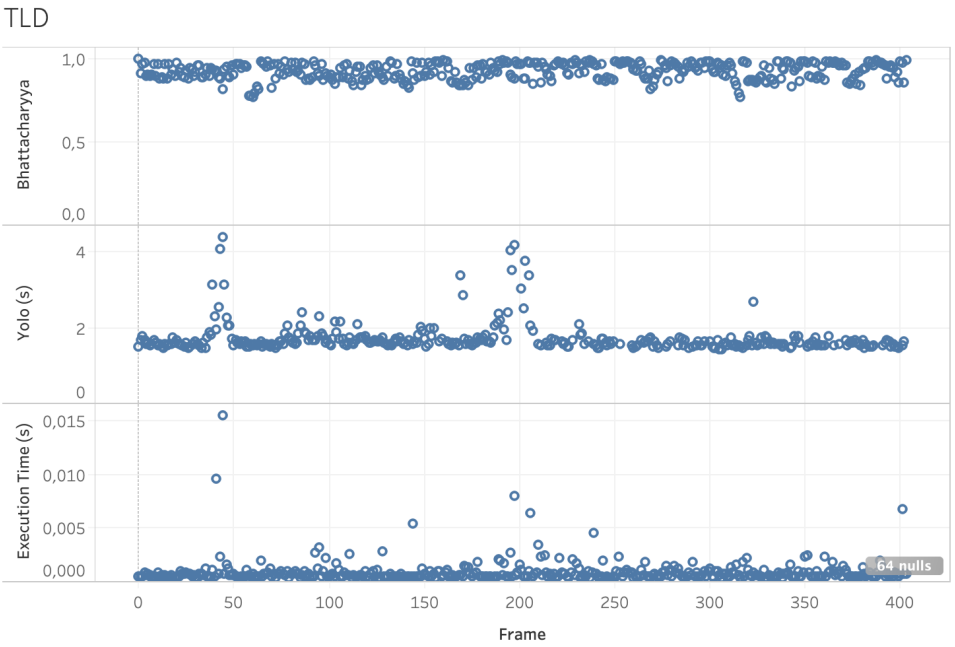


Figura 45 – Resultados utilizando o TLD e Bhattacharyya

7.6 EXPERIMENTO MEDIANFLOW

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *MEDIANFLOW (Fluxo médio)* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *MEDIANFLOW* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.6.1 MEDIANFLOW + Correlation

Na Figura 46, é possível observar no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

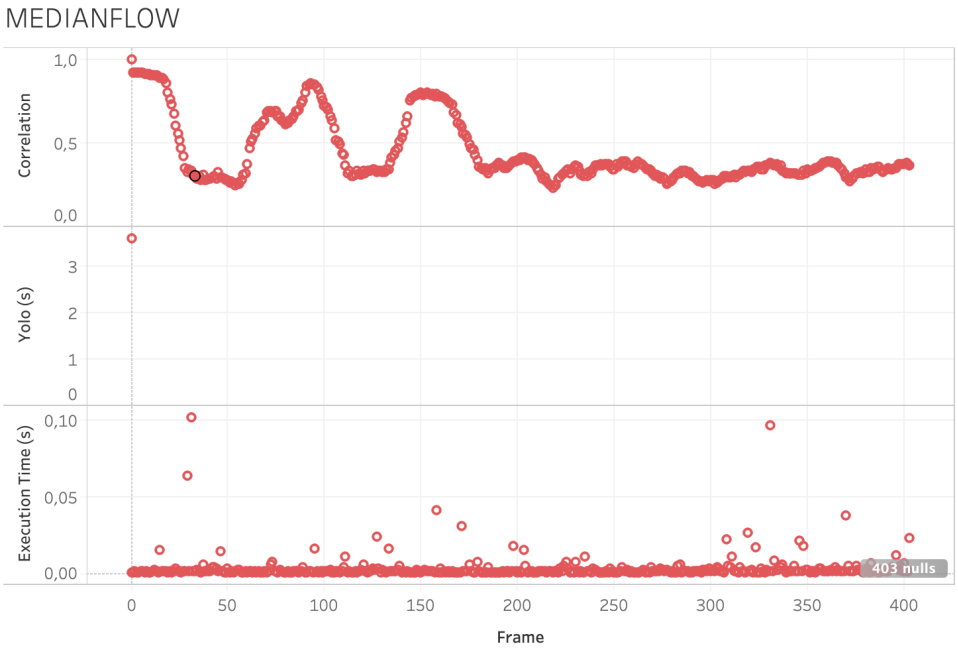


Figura 46 – Resultados utilizando o MEDIANFLOW e Correlation

7.6.2 MEDIANFLOW + Chi-Square

Na Figura 47 no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

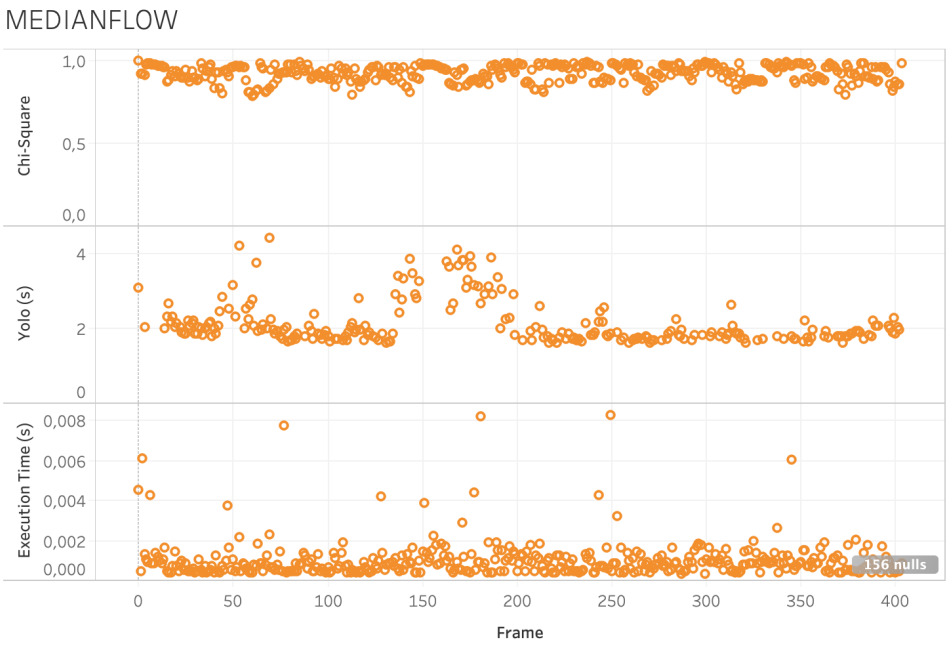


Figura 47 – Resultados utilizando o MEDIANFLOW e Chi-Square

7.6.3 MEDIANFLOW + Intersection

Na Figura 48 no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

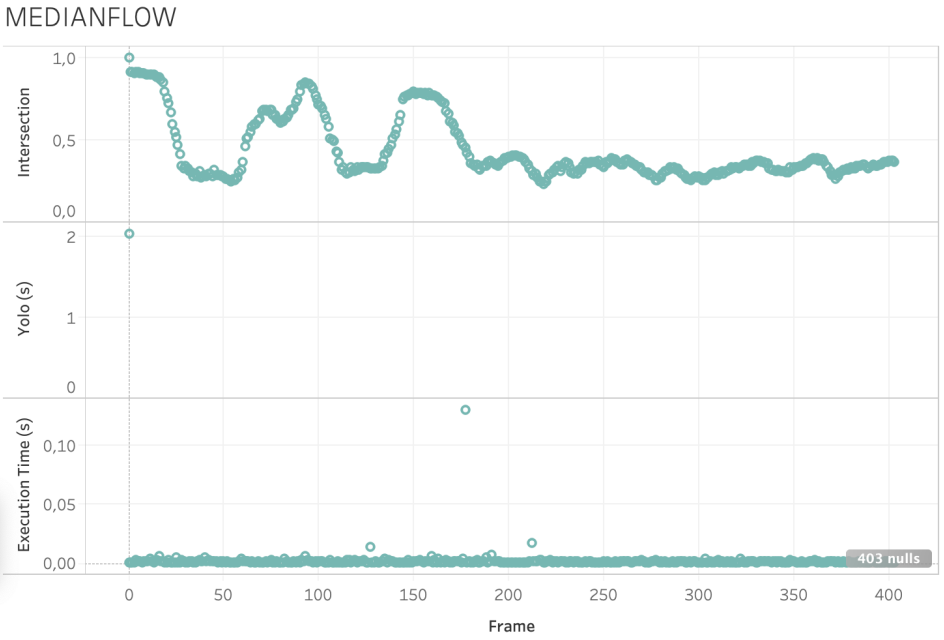


Figura 48 – Resultados utilizando o MEDIANFLOW e Intersection

7.6.4 MEDIANFLOW + Bhattacharyya distance

Na Figura 49, é possível observar no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

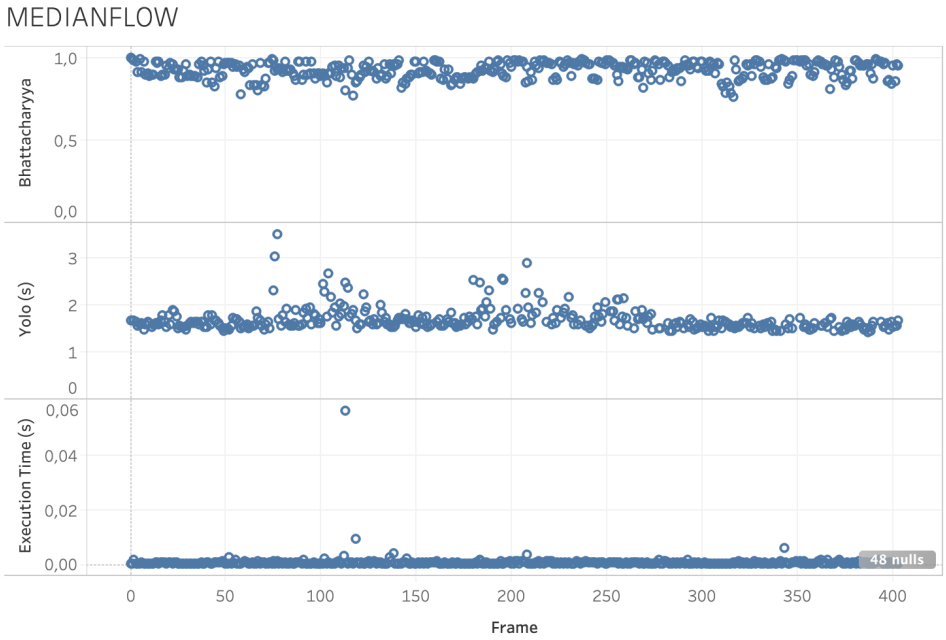


Figura 49 – Resultados utilizando o MEDIANFLOW e Bhattacharyya

7.7 EXPERIMENTO CSRT

Esta Seção apresenta os resultados obtidos pelos experimentos utilizando o algoritmo *YOLO* como parte da arquitetura *LIDA*. O método *CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability, ou DCF-CSR)* foi implementado como *codelet* de atenção dentro desta arquitetura cognitiva, afim de rastrear objetos previamente identificados pelo *YOLO*. Para isso, a seguir são apresentados diferentes experimentos com o rastreador de objetos *CSRT* junto de um segundo *codelet* de atenção, definido como *Change Detector* no Capítulo 6. O *codelet Change Detector* foi implementado com variações de métodos para a verificação de mudanças de cena (grandes diferença entre frames) durante a execução de videos.

7.7.1 CSRT + Correlation

Na Figura 50 é possível observar no quadrante *Correlation*, os resultados de correlação entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de correlação.

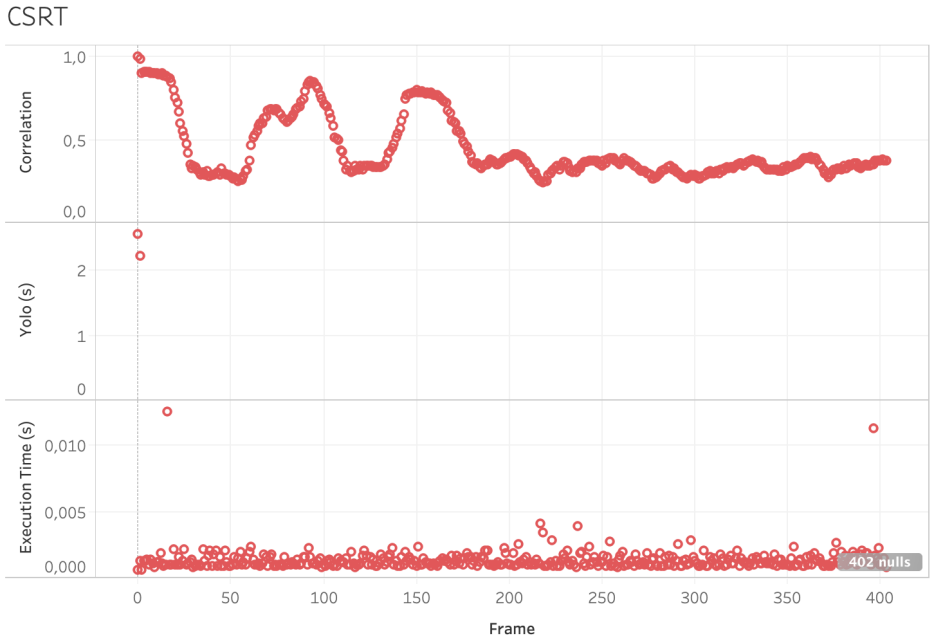


Figura 50 – Resultados utilizando o CSRT e Correlation

7.7.2 CSRT + Chi-Square

Na Figura 51 é possível observar no quadrante *Chi-Square*, os resultados do Chi-Quadrado entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo do Chi-Quadrado.

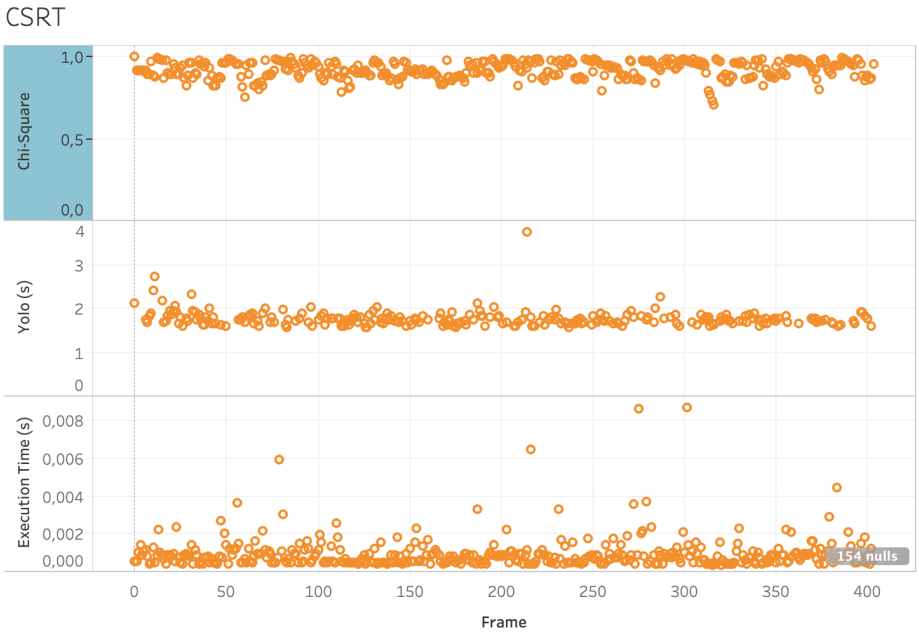


Figura 51 – Resultados utilizando o CSRT e Chi-Square

7.7.3 CSRT + Intersection

Na Figura 52 é possível observar no quadrante *Intersection*, os resultados de interseção entre todos os *frames* calculados. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo de interseção.

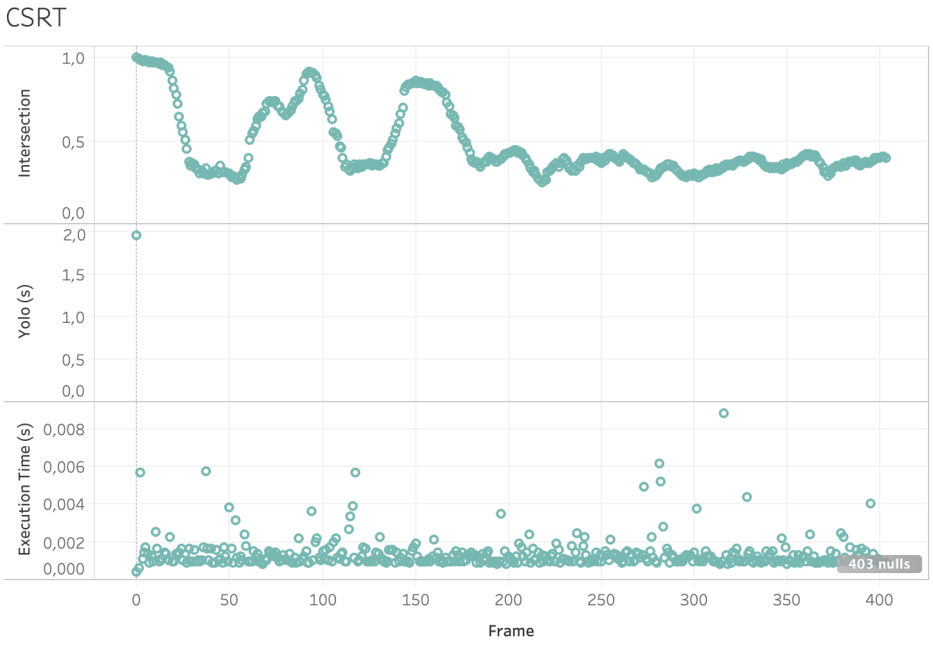


Figura 52 – Resultados utilizando o CSRT e Intersection

7.7.4 CSRT + Bhattacharyya distance

Na Figura 53 é possível observar no quadrante *Bhattacharyya*, os resultados da distância de *Bhattacharyya* calculada entre todos os *frames*. Já no quadrante *YOLO (s)*, pode ser observado o tempo de execução do *YOLO* na *PAM* e em quais quadros ele entrou em ação. Por fim, no quadrante *Execution Time (s)* são apresentados os tempos em segundos para o cálculo da distância de *Bhattacharyya*.

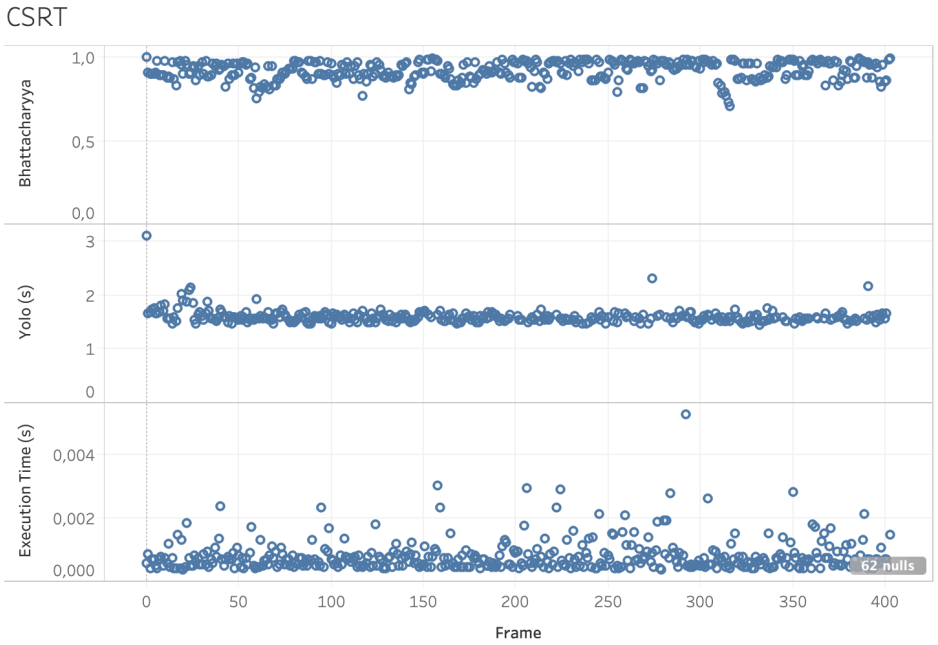


Figura 53 – Resultados utilizando o CSRT e Bhattacharyya

8 DISCUSSÃO E CONCLUSÃO

Neste Capítulo são apresentadas as discussões e conclusões sobre os resultados obtidos no Capítulo 7. As análises aqui apresentadas exploram os dados de diferentes formas, afim de entender de maneira mais profunda os principais fenômenos observados.

Nos experimentos realizados, é possível observar o desempenho de cada um dos algoritmos implementados como *codelet* de atenção, tanto o responsável pela verificação de mudança de cena, quanto o que realiza o rastreamento dos objetos sem a necessidade de um novo ciclo pela *PAM*.

8.1 CODELET DE ATENÇÃO PARA MUDANÇA DE CENA

Para todos os métodos testados como *codelet* de atenção para mudança de cena, foi apresentada uma maior utilização do *YOLO* quando utilizado o algoritmo de distância de *Bhattacharyya* (em azul), seguido pelo método *Chi*-quadrado (em laranja), como visto na Figura 54.

Ambos possuíram resultados semelhantes, variando apenas em relação aos algoritmos de rastreamento de objetos do segundo *codelet* de atenção observados a seguir.'

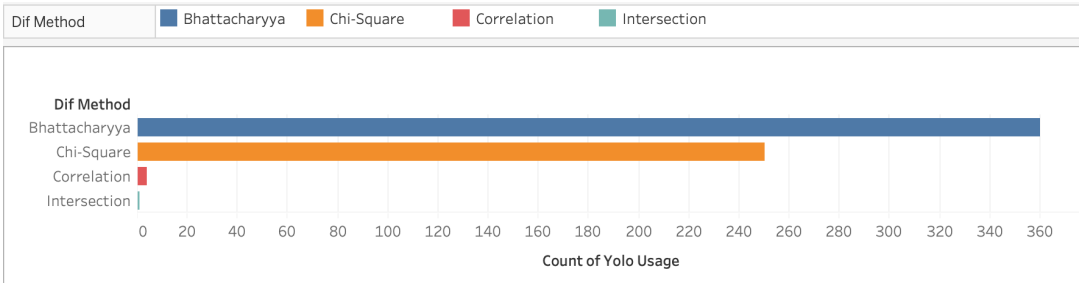


Figura 54 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos Boosting, totalizando a utilização do *YOLO* na *PAM*

Por outro lado, pode ser observado a baixa efetividade dos métodos de *correlação* e *interseção* ao calcularem as diferenças entre *frames*. Em ambos os casos, para um video com cerca de 400 *frames*, foram indicados menos de 10 momentos com mudança significativa na cena exibida (conforme a Figura 54). Esse efeito proporcionou uma alta velocidade no processo como um todo, porém os algoritmos de rastreamento de objetos não foram capazes de, sozinhos, manter um bom indice de assertividade.

8.2 DESEMPENHO DO BOOSTING

A maior presença da *PAM* representa um maior custo computacional com o uso constante do *YOLO*, porém é possível observar na Figura 55 que no total o *Chi*-quadrado foi mais custoso, tendo maior tempo gasto, mesmo com o método *Bhattacharyya* requisitando mais vezes a utilização da *PAM*. Isso se deve ao fato que os momentos indicados pelo *Bhattacharyya*, para um novo reconhecimento dos objetos, foram processados mais rapidamente pelo *YOLO*.

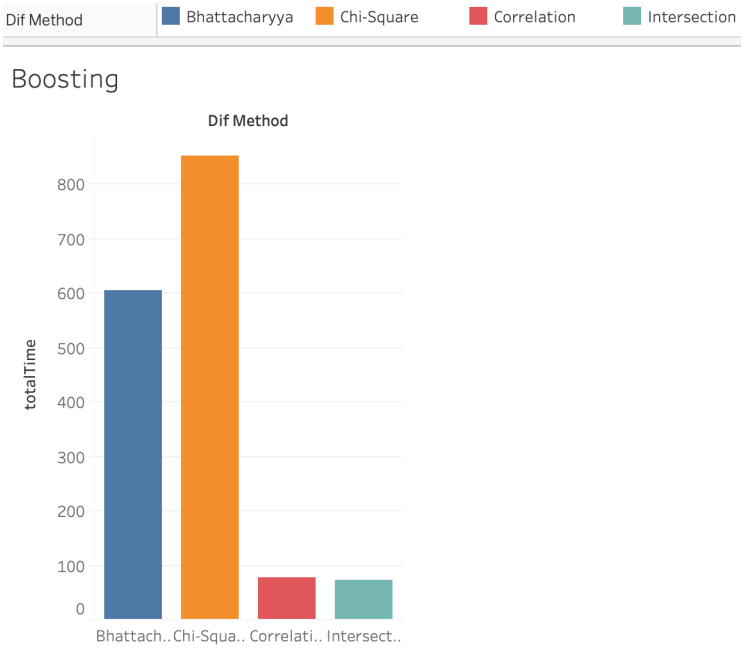


Figura 55 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos Boosting, totalizando o tempo de execução

Na Figura 56, no quadrante superior, é possível observar o baixo tempo gasto para cada um dos *codelets* de atenção compararem a diferença entre os *frames*. Já no segundo quadrante, é possível verificar claramente que o método *Bhattacharyya* indicou mais mudanças entre os frames do video, fazendo isso sobre quadros que o *YOLO* precisou de menos tempo para identificar os objetos.

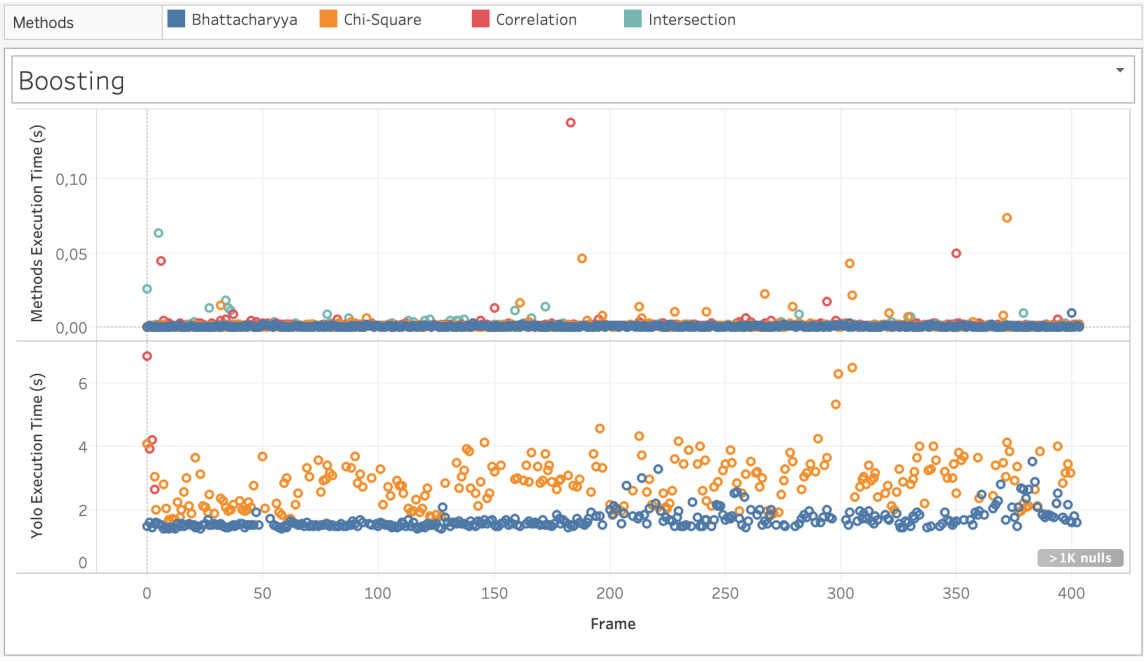


Figura 56 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos Boosting, para cada frame

O grau de assertividade pode ser observado na Figura 57, ficando evidente o melhor desempenho do *Boosting*, quando utiliza o método *Bhattacharyya* para identificar os momentos de mudança de cena. Em contrapartida, apenas para os primeiros frames onde os modelos de *correlação* e *interseçao* foram utilizados, que alguma assertividade foi obtida.

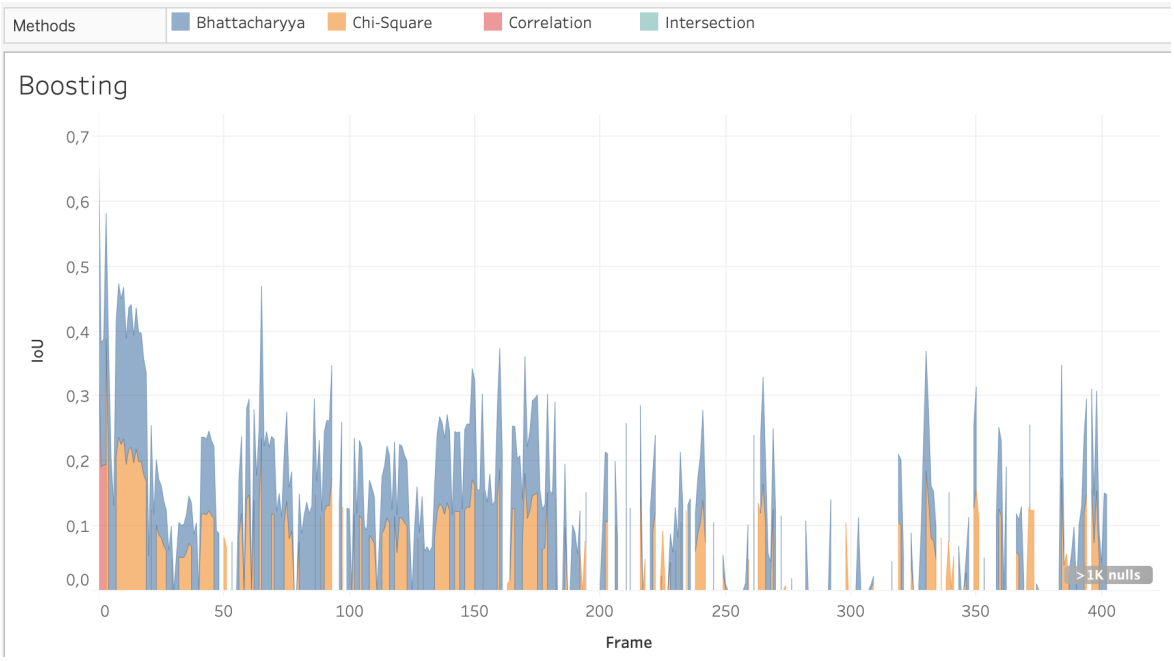


Figura 57 – Comparação do IoU de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos Boosting, para cada frame

8.3 DESEMPENHO DO CSRT

Aqui novamente, a maior presença da *PAM* representa um maior custo computacional com o uso constante do *YOLO*, porém ao contrário do ocorrido com o *Boosting* na Seção 8.2 é possível observar na Figura 58 que no total o *Bhattacharyya* agora foi mais custoso, tendo maior tempo gasto, que o método *Chi*-quadrado, vezes a utilização da *PAM*. Isso se deve ao fato que os momentos indicados pelo *Bhattacharyya*, para um novo reconhecimento dos objetos, foram processados mais rapidamente pelo *YOLO*.

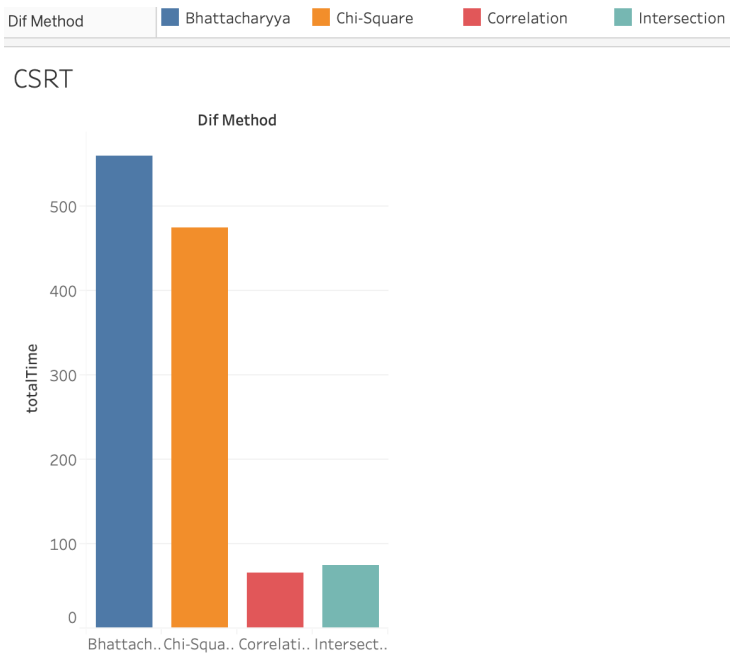


Figura 58 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos CSRT, totalizando o tempo de execução

Na Figura 59, no quadrante superior, é possível observar novamente o baixo tempo gasto para cada um dos *codelets* de atenção compararem a diferença entre os *frames*. Já no segundo quadrante, é possível verificar que o método *Bhattacharyya* selecionou quadros do video de entrada, ainda que menos custosos para o *YOLO* processar, tiveram um tempo médio de processamento mais do que os selecionados *Chi-quadrado*. Com isso, e por indicar um número maior de *frames* com mudança na cena, o modelo de *Chi-quadrado* foi superior ao ser utilizado com o *CSRT*.

Novamente, pode ser observado a baixa efetividade dos métodos de *correlação* e *interseção* ao calcularem as diferenças entre *frames*.

Figura 59 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos CSRT, para cada frame

O grau de assertividade pode ser observado na Figura 60, e mais uma vez o método *de distância de Bhattacharyya* obteve melhor desempenho para identificar os momentos de mudança de cena.

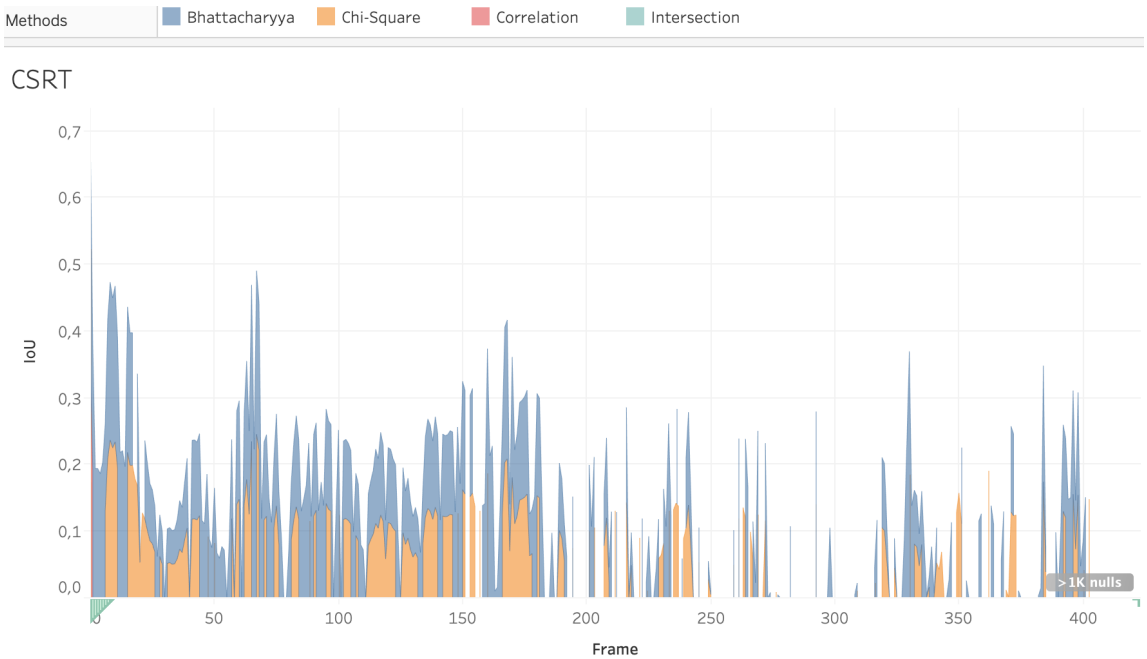


Figura 60 – Comparação do IoU de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos CSRT, para cada frame

8.4 DESEMPENHO DO KCF

O modelo de distância de *Bhattacharyya* ao ser utilizado em conjunto com o KCF, proporcionou uma maior presença da *PAM* em relação ao uso dos métodos observados anteriormente (Figura 62). O conjuto teve um impacto computacional ainda maior no tempo decorrido, cerca de 200 segundo a mais no total, em relação a utilização do *CSRT*. Neste caso, a distância de *Bhattacharyya* mesmo no geral teve um tempo superior ao *Chi*-quadrado como é possível observar na Figura 61.

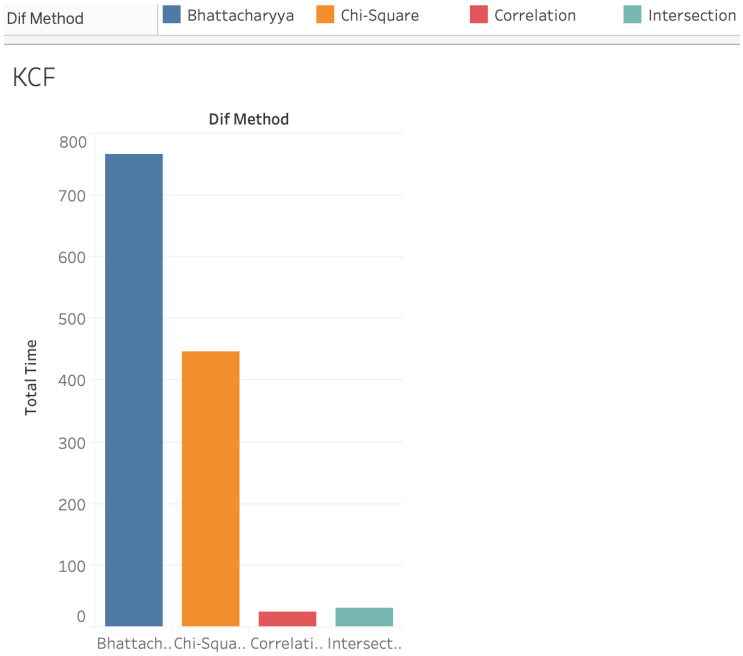


Figura 61 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos KCF, totalizando o tempo de execução

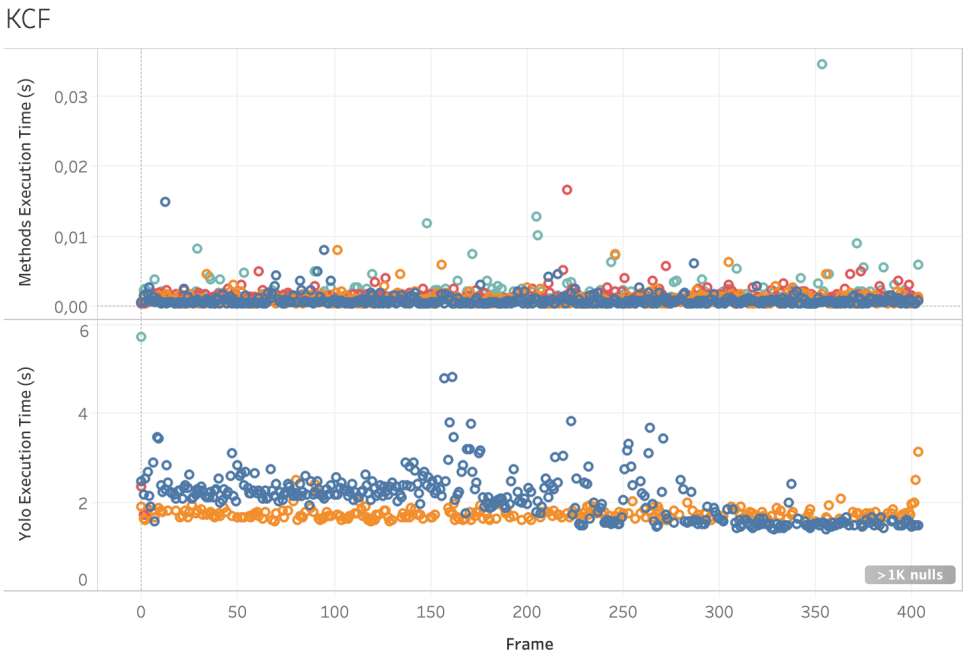


Figura 62 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos KCF, para cada frame

O grau de assertividade pode ser observado na Figura 63, neste caso, o modelo *de distância de Bhattacharyya* conseguiu picos de assertividade ainda maiores ao utilizar o KCF para seguir os objetos rastreados pelo *YOLO* na cena.

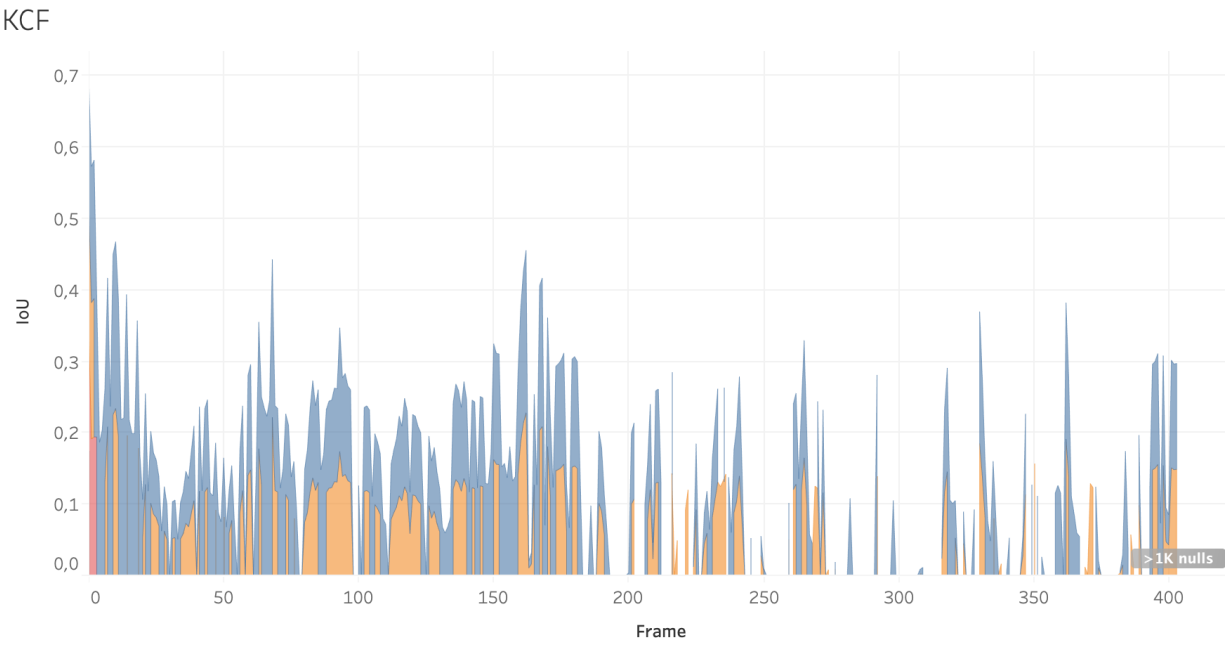


Figura 63 – Comparação do IoU de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos KCF, para cada frame

8.5 DESEMPENHO DO TLD

A utilização do TLD resultou em um empate técnico (Figura 64) entre o tempo total de execução para o *Chi*-quadrado e a distância de *Bhattacharyya*, com diferença de apenas $1{,}2s$ de vantagem para *Chi*-quadrado. Neste experimento, parte significativa dos *frames* selecionados pelo *Chi*-quadrado tiveram maior custo computacional para a *PAM*, fazendo com que seu tempo total de execução se equiparasse com o modelo em azul (Figura 65).

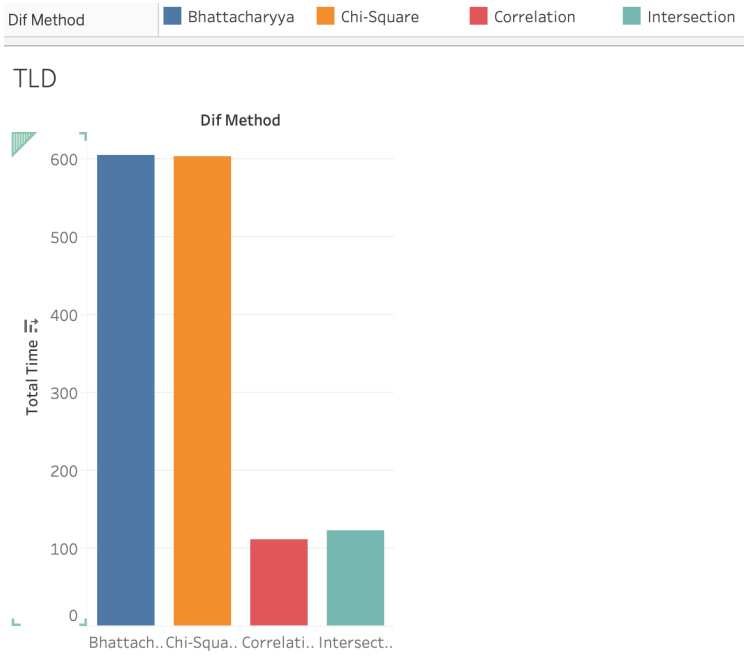


Figura 64 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos TLD, totalizando o tempo de execução

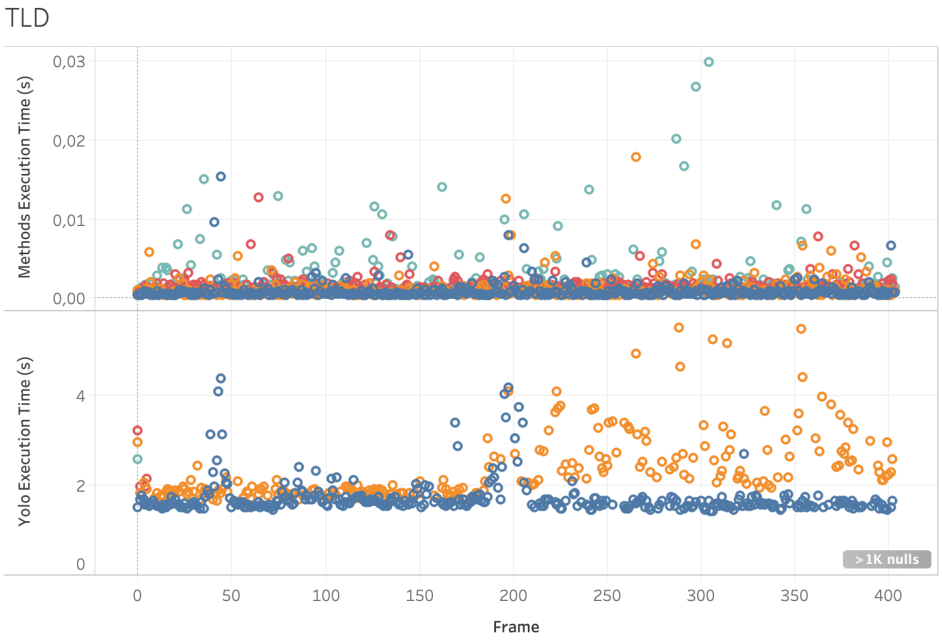


Figura 65 – Comparação de tempo de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos TLD, para cada frame

Como observado na Figura 66, modelo *de distância de Bhattacharyya* obteve resultados semelhantes aos obtidos com o uso do KCF, com picos de assertividade ainda elevados.

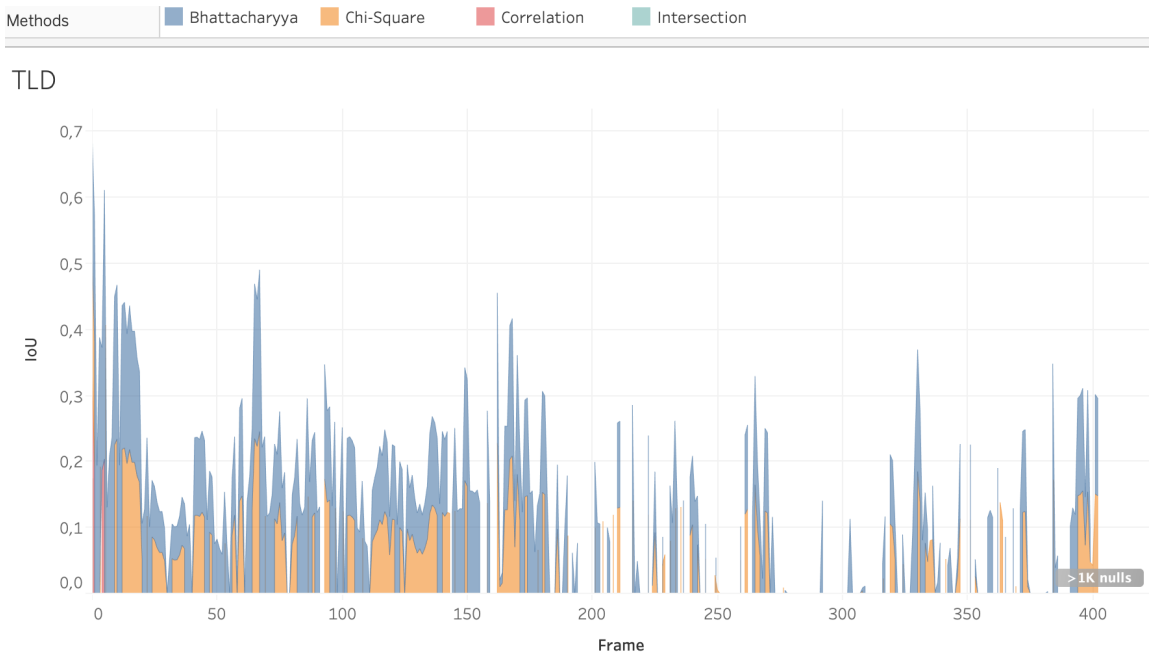


Figura 66 – Comparação do IoU de execução x frames, entre os diferentes métodos de compação de imagens utilizados com o rastreador de objetos TLD, para cada frame

8.6 CONCLUSÃO

Os experimentos realizados no Capítulo 7, demonstram que a arquitetura *LIDA* se apresenta como uma alternativa ao ser utilizada em cenários de reconhecimento de objetos quando existem hardwares limitados.

Foi possível diminuir o tempo de execução geral do processo de reconhecimento de objetos em cerca de 40%, como no exemplo de $19{,}5$ para aproximadamente $13s$ (conforme Figura 29 e Seção 7), sem abrir mão da utilização da versão da rede neural do *YOLO* com maior número de camadas, que possui melhor taxa de assertividade, mas que exige um hardware de maior capacidade quando utilizado fora da arquitetura *LIDA*. É possível afirmar, baseado nas discussões da Seção 8, que os algoritmos *Chi-quadrado* e *distância de Bhattacharyya*, utilizados como detector de diferença entre os quadros dos videos testados, selecionaram de forma mais assertiva os instantes em que se fez necessária a execução do *YOLO* na *PAM*.

Por sua vez, os rastreadores de objetos *CSRT*, *TLD* e *KCF* elevaram a assertividade média calculada pela *IoU* das *bound boxes* com o padrão-ouro, onde na amostra apresentada na Seção 8, mostrou *frames* com assertividades superiores aos obtidos pelo *YOLO* sem modificações, ao compararmos os resultados da Seção 7.1 com as Seções 7.4, 7.5 e 7.7. Assim, a hipótese de que os *codelets* de atenção rastreadores de objetos poderiam corrigir erros cometidos pelo *YOLO*, foi demonstrada empiricamente.

Os *codelets* de atenção se mostraram de suma importância para a realização de ajustes em tempo de execução, permitindo assim, a escolha do melhor momento para a execução de um novo reconhecimento de objetos, sem a necessidade de definir intervalos fixos de *frames* para tal tarefa. Além disso, os *codelets* de atenção abrem espaço para que nos próximos passos outros algoritmos sejam implementados no mesmo modelo, permitindo melhores ajustes em diferentes variáveis e avaliando cada situação específica em tempo de execução, como por exemplo, o limiar de certeza do reconhecimento de objetos realizado pelo *YOLO*, ou a escolha do melhor algoritmo para indicar mudança de cena a partir da classe de imagens processada.

REFERÊNCIAS

ANALIDE, C.; KIM, P. Experiential learning in data science: from the dataset repository to the platform of experiences. In: IOS PRESS. **Intelligent Environments 2017: Workshop Proceedings of the 13th International Conference on Intelligent Environments**. [S.l.], 2017. v. 22, p. 122.

Anderson, J. R. Language, memory, and thought. 1976.

ANDERSON, J. R. **How can the human mind occur in the physical universe?** [S.l.]: Oxford University Press, 2009.

ANDERSON, J. R.; LEBIERE, C. J. **The atomic components of thought**. [S.l.]: Psychology Press, 2014.

BAARS, B. J. In the theatre of consciousness. global workspace theory, a rigorous scientific theory of consciousness. **Journal of Consciousness Studies**, Imprint Academic, v. 4, n. 4, p. 292–309, 1997.

____. The conscious access hypothesis: origins and recent evidence. **Trends in cognitive sciences**, Elsevier, v. 6, n. 1, p. 47–52, 2002.

____. Global workspace theory of consciousness: toward a cognitive neuroscience of human experience. **Progress in brain research**, Elsevier, v. 150, p. 45–53, 2005.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. [S.l.]: "O'Reilly Media, Inc.", 2008.

CAMBRIA, E.; WHITE, B. Jumping nlp curves: A review of natural language processing research. **IEEE Computational intelligence magazine**, IEEE, v. 9, n. 2, p. 48–57, 2014.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. **Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on**. [S.l.], 2005. v. 1, p. 886–893.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**. [S.l.], 2009. p. 248–255.

DENŒUD, L.; GUÉNOCHE, A. Comparison of distance indices between partitions. In: **Data Science and Classification**. [S.l.]: Springer, 2006. p. 21–28.

DUDA, R. O.; HART, P. E. Pattern classification and scene analysis. **A Wiley-Interscience Publication, New York: Wiley, 1973**, 1973.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. **International journal of computer vision**, Springer, v. 88, n. 2, p. 303–338, 2010.

FRANKLIN, S. A conscious artifact? **Journal of Consciousness Studies**, Imprint Academic, v. 10, n. 4-5, p. 47–66, 2003.

FRANKLIN, S.; JR, F. P. The lida architecture: Adding new modes of learning to an intelligent, autonomous, software agent. **pat**, v. 703, p. 764–1004, 2006.

FRANKLIN, S.; KELEMEN, A.; MCCAULEY, L. Ida: A cognitive agent architecture. In: IEEE. **Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on**. [S.l.], 1998. v. 3, p. 2646–2651.

FRANKLIN, S. et al. Lida: A systems-level architecture for cognition, emotion, and learning. **IEEE Transactions on Autonomous Mental Development**, IEEE, v. 6, n. 1, p. 19–41, 2014.

____. A lida cognitive model tutorial. **Biologically Inspired Cognitive Architectures**, Elsevier, v. 16, p. 105–130, 2016.

____. Lida: A computational model of global workspace theory and developmental learning. 2007.

GAIKWAD, S. K.; GAWALI, B. W.; YANNAWAR, P. A review on speech recognition technique. **International Journal of Computer Applications**, International Journal of Computer Applications, 244 5 th Avenue,# 1526, New . . . , v. 10, n. 3, p. 16–24, 2010.

GAMEZ, D. Progress in machine consciousness. **Consciousness and cognition**, Elsevier, v. 17, n. 3, p. 887–910, 2008.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric environment**, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998.

GIRSHICK, R. Fast r-cnn. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1440–1448.

GIRSHICK, R. et al. Region-based convolutional networks for accurate object detection and segmentation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 38, n. 1, p. 142–158, 2015.

GÖK, S. E.; SAYAN, E. A philosophical assessment of computational models of consciousness. **Cognitive Systems Research**, Elsevier, v. 17, p. 49–62, 2012.

GRAZIANO, M. S. Speculations on the evolution of awareness. **Journal of cognitive neuroscience**, MIT Press, v. 26, n. 6, p. 1300–1304, 2014.

GRAZIANO, M. S.; WEBB, T. W. A mechanistic theory of consciousness. **International Journal of Machine Consciousness**, World Scientific, v. 6, n. 02, p. 163–176, 2014.

____. The attention schema theory: a mechanistic account of subjective awareness. **Frontiers in psychology**, Frontiers, v. 6, p. 500, 2015.

GROSSBERG, S. Competitive learning: From interactive activation to adaptive resonance. **Cognitive science**, Wiley Online Library, v. 11, n. 1, p. 23–63, 1987.

____. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. **Neural networks : the official journal of the International Neural Network Society**, v. 37, p. 1–47, 2013.

HE, K. et al. Mask r-cnn. In: IEEE. **Computer Vision (ICCV), 2017 IEEE International Conference on**. [S.l.], 2017. p. 2980–2988.

____. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

JAIN, A. K.; RATHA, N. K.; LAKSHMANAN, S. Object detection using gabor filters. **Pattern Recognition**, v. 30, p. 295–309, 1997.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Tracking-learning-detection. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 34, n. 7, p. 1409–1422, 2011.

KIERAS, D. E.; MEYER, D. E. An overview of the epic architecture for cognition and performance with application to human-computer interaction. **Human–Computer Interaction**, Taylor & Francis, v. 12, n. 4, p. 391–438, 1997.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

LAIRD, J. E.; NEWELL, A.; ROSENBLOOM, P. S. Soar: An architecture for general intelligence. **Artif. Intell.**, v. 33, p. 1–64, 1987.

LAIRD, J. E.; ROSENBLOOM, P. S.; NEWELL, A. Title chunking in soar : The anatomy of a general learning. In: . [S.l.: s.n.], 2007.

LANGLEY, P.; LAIRD, J. E.; ROGERS, S. Cognitive architectures: Research issues and challenges. **Cognitive Systems Research**, Elsevier, v. 10, n. 2, p. 141–160, 2009.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755.

LOPS, P.; GEMMIS, M. D.; SEMERARO, G. Content-based recommender systems: State of the art and trends. In: **Recommender systems handbook**. [S.l.]: Springer, 2011. p. 73–105.

LOWE, D. G. Object recognition from local scale-invariant features. In: **ICCV**. [S.l.: s.n.], 1999.

MADER, K. **TV77: Video Object Tracking**. 2018. Disponível em: <https://www.kaggle.com/kmader/videoobjecttracking>.

MAES, P. How to do the right thing. **Connection Science**, Taylor & Francis, v. 1, n. 3, p. 291–323, 1989.

MANZOTTI, R.; TAGLIASCO, V. Artificial consciousness: A discipline between technological and theoretical obstacles. **Artificial intelligence in medicine**, Elsevier, v. 44, n. 2, p. 105–117, 2008.

MILLER, G. A. The cognitive revolution: a historical perspective. **Trends in cognitive sciences**, Elsevier, v. 7, n. 3, p. 141–144, 2003.

NEWELL, A. **Unified theories of cognition**. [S.l.]: Harvard University Press, 1994.

NIETZSCHE, F.; KENNEDY, J. **The dawn of day**. [S.l.]: Courier Corporation, 2007.

PELLI, D. G.; TILLMAN, K. A. The uncrowded window of object recognition. **Nature neuroscience**, Nature Publishing Group, v. 11, n. 10, p. 1129, 2008.

PRASAD, D. K.; STARZYK, J. A. A perspective on machine consciousness. In: CITESEER. **Intl. Conf. Advanced Cognitive Technologies and Applications**. [S.l.], 2010. p. 109–114.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. **arXiv preprint**, 2017.

____. Yolov3: An incremental improvement. **arXiv preprint arXiv:1804.02767**, 2018.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 91–99.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

SAMSONOVICH, A. V. Toward a unified catalog of implemented cognitive architectures. **BICA**, v. 221, n. 2010, p. 195–244, 2010.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks : the official journal of the International Neural Network Society**, v. 61, p. 85–117, 2015.

SERMANET, P. et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. **arXiv preprint arXiv:1312.6229**, 2013.

STOCKMAN, G. C.; KOPSTEIN, S.; BENETT, S. Matching images to models for registration and object detection via clustering. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-4, p. 229–241, 1982.

SUN, R. Learning, action and consciousness: A hybrid approach toward modelling consciousness. **Neural Networks**, Elsevier, v. 10, n. 7, p. 1317–1331, 1997.

SUN, R.; BOOKMAN, L. A. **Computational architectures integrating neural and symbolic processes: A perspective on the state of the art**. [S.l.]: Springer Science & Business Media, 1994. v. 292.

SUN, R.; FRANKLIN, S. **Computational models of consciousness: A taxonomy and some examples.** [S.l.]: Cambridge University Press, 2007.

TANG, M. et al. Segmentation-by-detection: A cascade network for volumetric medical image segmentation. In: IEEE. **Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on**. [S.l.], 2018. p. 1356–1359.

TONONI, G. The integrated information theory of consciousness: an updated account. **Archives italiennes de biologie**, v. 150, n. 2/3, p. 56–90, 2011.

UIJLINGS, J. R. et al. Selective search for object recognition. **International journal of computer vision**, Springer, v. 104, n. 2, p. 154–171, 2013.

VELMANS, M. How to define consciousness: And how not to define consciousness. **Journal of Consciousness Studies**, Imprint Academic, v. 16, n. 5, p. 139–156, 2009.

ZENG, X. et al. Segmentation and measurement of the cortex from 3-d mr images using coupled-surfaces propagation. **IEEE transactions on medical imaging**, IEEE, v. 18, n. 10, p. 927–937, 1999.