RELATÓRIO PARCIAL

Processos atencionais e aprendizado de máquina para sistemas robóticos

Aluno: Erik de Godoy Perillo
Orientadora: Profa. Dra. Esther Luna Colombini

Instituto de Computação
Universidade Estadual de Campinas

9 de dezembro de 2016

I. Introdução

A capacidade de percepção e construção de um modelo da realidade ao seu redor é fundamental para que sistemas robóticos interajam com o ambiente e executem tarefas diversas e complexas que podem ter as mais variadas utilidades para os humanos. Um componente fundamental para isso é a habilidade de dar foco apenas ao relevante, evitando assim o processamento desnecessário de enormes quantias de dados.

A atenção é um processo que faz parte do dia a dia de diversos seres vivos em diversas maneiras e é razoável inspirar-se nela para a construção de mecanismos semelhantes para a construção de sistemas de inteligência artificial em máquinas. Tal área tem sido foco de estudo há anos, resultando em diversas teorias em psicologia sobre a atenção humana que inspiraram a implementação de modelos computacionais bem sucedidos.

Neste trabalho, objetivamos construir um modelo atencional eficiente. Com base no modelo, implementaremos um *framework* atencional para robôs móveis que permita o uso da seleção em tempo real dos estímulos mais relevantes para as mais diversas tarefas que o robô possa executar. No trabalho atual focamos na atenção visual, mas o objetivo final do sistema é que ele funcione para outros sensores. O *framework* também contará com um módulo de reconhecimento de objetos que poderá ser substituído.

A. Objetivos da primeira parte do projeto

Os objetivos principais para o primeiro semestre do trabalho eram:

- Revisão bibliográfica sobre teorias sobre a atenção e diversos modelos.
- Escolha das técnicas mais adequadas para o processo atencional e o reconhecimento de objetos.
- Implementação de um modelo atencional.

Há uma quantidade surpreendente de avanços recentes na área de modelos de saliência visual. Entender os avanços mais relevantes é importante para a obtenção de um sistema atencional eficiente, então foi requerido mais tempo que o previsto para essa parte. Assim, todas as etapas previstas tiveram avanço, com exceção da parte de reconhecimento de objetos, a qual optamos por deixar para mais tarde pois a mesma serve como um complemento para nosso trabalho e é de menor relevância que o componente atencional. As atividades desenvolvidas são mais detalhadas a seguir.

II. Resumo das atividades

A. Revisão Bibliográfica

Dois dos conceitos importantes para o entendimento da literatura do meio são:

- *features*: Características básicas que formam entidades visuais. Podem ser de várias classes, como cor (verde, azul), orientação (horizontal, vertical), luminância, tamanho.
- *Bottom-up vs. Top-down*: Por componente *bottom-up* de atenção entende-se saliências instintivas percebidas por mudanças e/ou contrastes muito grandes em uma cena. O componente *top-down* é aquele que dá saliência variável às *features* de acordo com a meta do agente do momento.

A maioria dos modelos computacionais baseia-se em teorias formadas na psicologia. Duas das mais famosas são a *Feature Integration Theory* (FIT) [15] e a *Guided Search* [5]. Ambas provêm contribuições importantes para o entendimento dos processos de saliência visual. Diversos modelos computacionais foram criados baseando-se em ideias delas. Começou-se então por elas.

A FIT indica basicamente que se a busca de um objeto de interesse em uma cena for por apenas uma *feature*, a localização é feita em tempo instantâneo. Entretanto, se o objeto de interesse for composto por múltiplas *features* a serem buscadas (e. g. uma linha horizontal verde), a localização do objeto é feita em tempo linear.

Já Guided Search diz que buscas por conjunções de *features* são na verdade mais rápidas pois a combinação das features gera um sinal de saliência mais forte no campo visual humano.

O VOCUS [4] é um modelo atencional computacional para a detecção de saliências visuais. A maioria dos seus componentes é feita com base nas ideias da FIT. Nesta primeira etapa, exploramos seu componente *bottom-up*. Ele lida com as *features*: cor, intensidade, orientação. Seus mapas de saliência são calculados com base nessas *features* e em diversas dimensões da imagem.

A revisão bibliográfica feita mostrou-se fundamental para os trabalhos desenvolvidos no semestre.

B. Modelo atencional

A carga teórica adquirida foi útil para a concepção do nosso modelo, chamado de *att*. Muitos mecanismos foram inspirados no VOCUS, lidando com as mesmas *features* e em múltiplas dimensões da imagem.

1) Extração de features: O modelo extrai os seguintes mapas de uma certa imagem: luminância, luminância invertida, vermelho, verde, amarelo, azul, orientações vertical, horizontal, 45°e 135°. Os de luminância e cor podem ser extraídos pela conversão da imagem para o espaço de cor LAB e as orientações são extraídas usando-se filtros de Gabor.

2) Extração de saliência: Para cada mapa, a saliência é calculada usando-se o mecanismo de *center-surround* [4]: uma operação que basicamente extrai contrastes fortes do mapa, dando intensidades de pixel altas

para essas regiões. O *center-surround* pode ser aplicado por meio da convolução de um *kernel* apropriado na imagem. Tal *kernel* pode ter tamanho variado, mas a soma de seus valores deve ser zero, o centro deve ter positivo e a vizinhança deve ter valores negativos.

Figura 1. À esquerda, a imagem original. No centro, seu mapa de oponência amarelo-azul. À direita, o resultado de *center-surround* no mapa de cor.

3) Mapas de saliência para cada feature: Para cada *feature* (e. g. vermelho) é calculado o *center-surround*. Isso é feito na imagem original e em diversas outras dimensões dela, calculando-se a pirâmide da imagem. Geralmente usa-se quatro níveis. Isso é importante para capturar saliências nos mais diversos níveis de detalhe da imagem. Uma vez calculados, todos os mapas de uma certa *feature* são redimensionados para as dimensões originais e somados, formando assim um mapa de *feature*.

4) Normalização: Uma vez calculados os mapas para cada *feature* (vermelho, orientação horizontal etc), é preciso fazer uma normalização nos mesmos. Isso se deve ao fato que, se há grande frequência de picos de saliência no mapa de vermelho, por exemplo, este não é de muito valor, pois o que se quer identificar são regiões salientes com relação à imagem como um todo. Assim, para cada mapa é calculado um peso de normalização.

São diversos os critérios desenvolvidos, como: número de máximos locais, densidade de máximos locais, espalhamento espacial dos máximos. Uma análise das alternativas não mostrou muitas diferenças no desempenho, então opta-se pelo método mais simples, por padrão, que é o número de máximos locais. Isso pode ser obtido por limiar *Otsu*, seguido de um algoritmo de componentes conexos para contar os máximos locais.

5) Mapas de saliência para cada feature: Uma combinação hierárquica dos mapas é feita após as normalizações. No final dessa operação, há três mapas: cor, luminância e orientação. Eles são formados simplesmente somando e normalizando suas instâncias: O de cor, por exemplo, é formado somando-se os mapas de saliência de vermelho, verde, amarelo e azul.

6) Combinação final: É dado um peso para cada mapa de saliência (cor, luminância, orientação) e então eles são somados e normalizados. Nos testes feitos, obteve-se melhores resultados dando peso maior para a cor e menor para a orientação. Nos exemplos aqui, os pesos são 2, 1, 0.1 para cor, luminância e orientação, respectivamente.

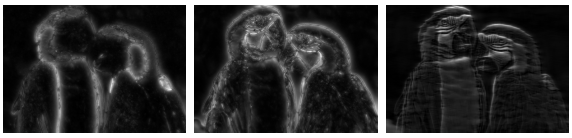

Figura 2. Mapas de saliência da figura original 1. Esquerda: mapa de cor. Centro: mapa de luminância. Direita: mapa de orientação.

Figura 3. Mapa de saliência final para uma figura. À esquerda, a imagem original. À direita, o mapa de saliência final.

7) Implementação: Todas as etapas do modelo descritas aqui foram implementadas. A linguagem utilizada foi Python, usando-se OpenCV e numpy. O código está disponível de modo público [12].

8) Considerações: O modelo implementado foi testado em diversas imagens e dá resultados satisfatórios na maioria dos casos: em regiões intuitivamente mais salientes, o mapa mostra a região como mais clara (como na figura 3). Há muitos hiperparâmetros para o modelo, como: níveis de pirâmide, valores dos filtros de Gabor, método de normalização, pesos para os mapas. Isso exige buscas exaustivas no espaço de alta dimensionalidade dos hiperparâmetros para achar bons valores, o que é muito custoso. Assim, embora o modelo atual dê bons resultados, um ponto negativo é a alta quantidade de hiperparâmetros.

C. Avaliação de desempenho

Foi feita uma pesquisa das métricas e *datasets* mais usados para avaliação dos modelos computacionais, pois isso é muito importante para o avanço do trabalho.

1) Datasets: Os *datasets* são compostos de imagens juntamente ao conjunto-verdade, algo que indica como os mapas de saliência para cada imagem deveriam ser.

Dois dos tipos principais de mapas-verdade:

1) Pontos de fixação de olhar: Algum dispositivo de *eye-tracking* é usado em pessoas para determinar as regiões na imagem onde as mesmas olham em um intervalo de tempo. Apenas os pixels com fixação são brancos.
2) Máscaras contínuas: As regiões salientes têm valores contínuos em um intervalo de intensidade de *pixels*, com valores mais altos representando uma saliência maior. São geralmente obtidas através dos pontos de fixação, aplicando-se uma gaussiana adequada em cada ponto.

Alguns *datasets* utilizados geralmente:

- `CAT2000 [1]`: 4000 imagens de 20 categorias com dados de fixação de olhar de 24 pessoas por imagem de duração de 5 segundos.
- `JUDD [9]`: 1003 imagens de diversos tipos com dados de fixação de olhar de 15 pessoas por imagem. Acompanha também mapas contínuos. É amplamente usado.
- `MIT300 [10]`: 300 imagens de diversos tipos com dados de fixação. Entretanto, os mesmos não estão disponíveis e devem ser testados pelos donos do *benchmark*.

2) Métricas: Muitas das métricas aqui mostradas são discutidas em [3], onde a aplicabilidade, significado, vantagens e desvantagens de cada uma são discutidas mais a fundo. Aqui daremos apenas uma breve descrição das mesmas.

- `AUC-Judd`
  Abreviação de *Area Under ROC Curve*. É feita uma varredura em diversos valores de *threshold* para o mapa de saliência e, para cada valor, calcula-se o *true positive rate* e o *false positive rate* com base na máscara feita para o mapa de saliência e os pixels brancos respectivos aos pontos de fixação de humanos. A área embaixo da *ROC* formada então é calculada. Usado em [10, 9].
- `NSS`
  `AUC` pode avaliar com altas pontuações mapas com falsos positivos mas com valores baixos nestes falsos positivos [3]. O `NSS` penaliza isso e dá altas pontuações a mapas com valores baixos em negativos e altos em positivos:

$$NSS(P,Q) = \frac{1}{N}\sum_{i=1}^{N} \bar{P}_i Q_i$$

  Onde N é o número de pontos de fixação, Q é o mapa binário de pontos de fixação e $\bar{P}$ é o mapa de saliência normalizado pelo desvio padrão. Usado em [10].
- `Similarity`
  Métrica para mapas contínuos. Definido como:

$$SIM(P,Q) = \frac{1}{N}\sum_{i=1}^{N} min(P_i, Q_i)$$

  Com P e Q indo de 0 a 1. Um valor de 1 define mapas idênticos e 0 mapas totalmente diferentes. Ambos os mapas são normalizados pela soma de cada um. Usado em [10].
- `Correlation Coefficient`
  Métrica para mapas contínuos. `Similarity` penaliza *false negatives* mais que *false positives* [3]. A métrica `CC` trata os dois simetricamente. Dada por:

$$CC(P,Q) = \frac{cov(P,Q)}{\sigma(P)\sigma(Q)}$$

  Onde P é normalizado no intervalo $[0, 1]$. Usado em [10].

Para *ground-truth* por pontos binários de fixação, `AUC` é razoável para casos em que se quer um bom *recall* e que falsos negativos mas com valores relativamente baixos não importam muito. Isso é verdade em casos em que o foco atencional será direcionado apenas à região de maior valor. `NSS` também pode ser válido pois ele penaliza falsos positivos mesmo que baixos, focando mais em *precision*. Para *ground-truth* contínuos derivados de pontos de fixação, `SIM` é bom quando importa-se com *recall*, pois a métrica penaliza falsos negativos mais que falsos positivos. `CC` também é boa por ser neutra e penalizar falsos positivos/negativos igualmente.

3) Comparação com outros modelos: Foi construído um ambiente de *benchmark* para o modelo *att*, que permite fazer testes automatizados com diversos *datasets* e métricas. O primeiro uso deste ambiente foi para fazer uma busca inicial de hiperparâmetros para o modelo: muitos valores destes diminuíam muito o desempenho do modelo, alguns aumentando o valor em uma métrica mas diminuindo consideravelmente em outra. Assim, foi feita uma escolha de hiperparâmetros que não prejudicasse muito nenhuma métrica.

O conjunto de imagens escolhido foi `Judd` [9]. Com uma amostragem aleatória de 128 imagens, calculamos diversas métricas e comparamos com modelos presentes na tabela do *MIT Saliency Benchmark* [2], um *benchmark* amplamente utilizado. Na comparação aqui mostrada, comparamos nosso modelo com três tipos de modelos: um que fica consistentemente entre os primeiros para diversas métricas (*DeepFix* [7]), um que fica consistentemente na metade (*Rosin Saliency 2* [13]), e um que fica consistentemente entre os últimos (IttiKoch [14]). Usamos duas métricas para pontos de saliência (AUC Judd e NSS) e duas para mapas contínuos (SIM e CC).

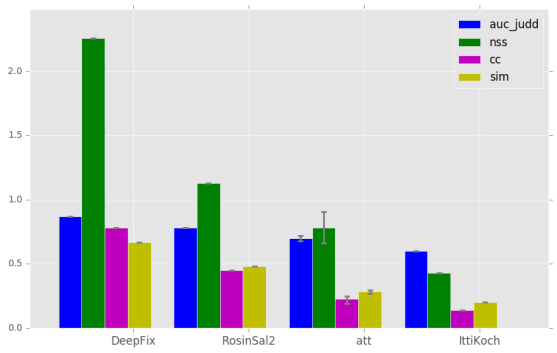

Figura 4. Comparação de desempenho do modelo att com diversos modelos do *MIT Saliency Benchmark*

Nota-se que nosso modelo é superior em todas as métricas ao *IttiKoch*, tem resultados comparáveis (mas sempre piores) ao *Rosin Saliency 2*, e tem resultados muito inferiores aos do *DeepFix*. Modelos modernos do topo como o *DeepFix* usam técnicas muito diferentes das usadas pelo nosso modelo, muitas vezes usando *Deep Learning*. Isso motiva a busca de uma nova abordagem para o nosso modelo.

III. Próximos passos

A. Nova versão do modelo att

Alguns dos modelos computacionais de saliência visual que ficam consistentemente no topo da lista do MITSAL são: *DeepFix* [7], *Salicon* [6], *Deep Gaze 2* [8], *SalNet* [11]. O que todos estes têm em comum é o uso de técnicas de *Deep Learning*, com o uso de redes neurais convolucionais. Ao que tudo indica, modelos usando tais técnicas têm resultados sempre superiores aos modelos como o VOCUS e a maior parte da comunidade na área adotou essa abordagem.

Um próximo passo natural para o projeto é aplicar técnicas semelhantes às dos modelos dominantes. Uma pesquisa nos modelos do topo foi feita para analisar melhor suas técnicas e avaliar a possibilidade da implementação de suas ideias. Com base na boa descrição do modelo e os excelentes resultados, escolheu-se basear no *DeepFix* para a nova versão do modelo att. Planeja-se implementar o modelo como um todo – ou sua maior parte –.

1) Extensão para vídeo: Não há sinal na literatura de algum modelo que foi projetado para funcionar também em vídeos. A extensão de modelos para vídeos não é imediata, pois deve-se contar com efeitos como a transição de foco de atenção ao longo do tempo, um efeito que não existe em imagens. Assim, faz parte dos próximos passos desenvolver um modo de estender o modelo novo do att – provavelmente baseado no *DeepFix* – para análise atencional em vídeos. Tal contribuição seria inédita até o momento para a área.

B. Construção Framework

A etapa final é a construção de um *Framework*. O mesmo será implementado em um robô móvel, onde serão definidas tarefas de navegação e será feita uma avaliação do desempenho do robô.

Referências

[1] Ali Borji e Laurent Itti. “CAT2000: A Large Scale Fixation Dataset for Boosting Saliency Research”. Em: *CVPR 2015 workshop on "Future of Datasets"* (2015).

[2] Zoya Bylinskii et al. *MIT Saliency Benchmark*.

[3] Zoya Bylinskii et al. “What do different evaluation metrics tell us about saliency models?” Em: (2016).

[4] Simone Frintrop. “Vocus: a visual attention system for object detection and goal-directed search”. Tese de doutorado. 2006.

[5] Susan L. Franzel Jeremy M. Wolfe Kyle R. Cave. “Guided Search: an alternative to the feature integration model for visual search”. Em: (1989).

[6] Ming Jiang et al. “SALICON: Saliency in Context.” Em: *CVPR*. 2015.

[7] Srinivas S. Kruthiventi, Kumar Ayush e R. Venkatesh Babu. “DeepFix: A Fully Convolutional Neural Network for predicting Human Eye Fixations”. Em: *CoRR* (2015).

[8] Matthias Kümmerer, Thomas S. A. Wallis e Matthias Bethge. “DeepGaze II: Reading fixations from deep features trained on object recognition”. Em: *CoRR* (2016).

[9] *Learning to predict where people look*. 2016. URL: http : / / people . csail . mit . edu / tjudd / WherePeopleLook / index . html (acesso em 04/10/2016).

[10] *MIT saliency benchmark*. 2016. URL: http : / / saliency . mit . edu / index . html (acesso em 27/09/2016).

[11] Junting Pan et al. “Shallow and Deep Convolutional Networks for Saliency Prediction”. Em: *CoRR* abs/1603.00845 (2016).

[12] *Repositório att*. 2016. URL: https://github.com/erikperillo/att (acesso em 09/12/2016).

[13] Paul L. Rosin. “A Simple Method for Detecting Salient Regions”. Em: *Pattern Recogn.* 42.11 (nov. de 2009).

[14] *Salicency Toolbox*. URL: http : / / www . saliencytoolbox.net/.

[15] A M Treisman e G Gelade. “A feature-integration theory of attention”. Em: *Cognit Psychol* 12.1 (jan. de 1980), pp. 97–136.