

Universidade de São Paulo  
Instituto de Matemática e Estatística  
Matemática Aplicada - Bacharelado - Habilitação em Métodos Matemáticos

Rafael Pereira Lima

# **Rastreamento de Olhar Usando Compressive Sensing**

São Paulo  
Dezembro de 2016

# Rastreamento de Olhar Usando Compressive Sensing

Monografia final da disciplina  
MAP2080 – Trabalho de Formatura.

Supervisor: Prof. Dr. Carlos Hitoshi Morimoto

São Paulo

Dezembro de 2016

# Resumo

Informações do olhar podem ser usadas para possibilitar a interação com o computador quando, por limitações físicas, a pessoa é incapaz de interagir com dispositivos de entrada usuais. Rastreamento de olhar consiste na identificação da posição do olhar durante determinado período. Desenvolvemos um programa de rastreamento de olhar usando *Compressive Sensing*, uma técnica usada para reconstruir sinais a partir de poucas amostras. Os resultados obtidos foram semelhantes a sistemas comerciais e outros presentes na literatura.

**Palavras-chave:** *Compressive Sensing*, rastreamento de olhar, processamento de imagens.

# Abstract

Gaze information can be used to enable human-computer interaction when the person is unable to use common input devices due to physical disabilities. Gaze tracking is the gaze position estimation in some period of time. We developed a gaze tracking algorithm using Compressive Sensing, a technique used to recover signals from few samples. The results are similar to commercial systems and those in the literature.

**Keywords:** Compressive Sensing, gaze tracking, image processing.

# Sumário

<b>1</b>	<b>Introdução</b>	<b>7</b>
<b>2</b>	<b>Rastreamento de olhar</b>	<b>9</b>
2.1	Aplicações . . . . .	9
2.2	Técnicas de rastreamento de olhar . . . . .	10
2.3	Características relevantes . . . . .	14
<b>3</b>	<b><i>Compressive Sensing</i></b>	<b>15</b>
3.1	Unicidade de $(P_0)$ . . . . .	20
3.2	Unicidade de $(P_1)$ . . . . .	23
3.3	Equivalência entre $(P_0)$ e $(P_1)$ . . . . .	26
3.4	Matrizes que satisfazem as condições de CS . . . . .	32
<b>4</b>	<b>O algoritmo de homotopia</b>	<b>34</b>
4.1	Exemplo . . . . .	38
<b>5</b>	<b>O modelo <i>cross-and-bouquet</i></b>	<b>40</b>
5.1	Desempenho . . . . .	42
<b>6</b>	<b>Desenvolvimento de um rastreador de olhar usando CS</b>	<b>44</b>
6.1	Coleta de amostras . . . . .	45

6.2	Rastreamento de olhar . . . . .	46
6.3	Resultados . . . . .	48
<b>7</b>	<b>Conclusão</b>	<b>51</b>

# Capítulo 1

## Introdução

Um dos problemas enfrentados por pessoas com deficiência motora é a falta de acesso à informação devido à impossibilidade ou dificuldade para usar dispositivos eletrônicos. Uma possível solução para esse problema seria usar o movimento dos olhos para interagir com o mundo exterior. O uso de técnicas para identificar onde determinada pessoa está olhando é chamado de rastreamento de olhar.

Para rastrear o olhar geralmente uma câmera é posicionada na frente de um dos olhos, como mostra a Figura 1.1. A partir da imagem obtida pela câmera, técnicas de processamento de imagens são usadas para identificar a posição da pupila e, a partir disso, identificar a direção em que o usuário está olhando. No entanto, o rastreamento de olhar é dificultado por vários fatores, como mudanças na iluminação do lugar e oclusão parcial dos olhos causada por alguma expressão facial. Portanto, não é uma tarefa trivial mas, apesar dessas dificuldades, deve ser executada em tempo real.

Em vez de localizar a pupila na imagem, podemos comparar cada nova imagem do olho com as amostras, ou seja, com imagens do olho correspondendo a diferentes posições de olhar registradas durante a calibração. Podemos encontrar as amostras mais parecidas com a imagem e depois estimar o olhar como uma média ponderada do olhar nas amostras.

Uma imagem em escala de cinza do olho pode ser escrita como uma combinação linear



Figura 1.1: Exemplo de câmera usada para rastreamento de olhar. Uma câmera posicionada na frente do olho direito registra imagens do olho, que serão usadas para estimar o olhar. Reproduzido de <https://pupil-labs.com/blog/2016-07/new-headset-color/>

das amostras mais um ruído, representado como uma combinação linear da base canônica de espaço vetorial (onde cada elemento da base pode ser interpretado como uma imagem que é preta em todos os *pixels*, exceto em um). Note que a combinação linear não é única, pois é possível calcular qualquer vetor como combinação linear dos elementos da base.

Quando a imagem é muito mais parecida com as amostras do que qualquer vetor da base, assumimos que ela pode ser escrita como uma combinação linear das amostras mais o ruído, onde os coeficientes do ruído são próximos de zero.

Podemos então procurar o vetor de coeficientes com o maior número de coeficientes nulos. Essa é a abordagem sugerida pelo Compressive Sensing (CS). Chamamos de esparso um sinal que depende apenas de uma pequena quantidade de vetores da base. Quando o sinal depende de vários vetores, chamamos de *compressive*.

Desenvolvemos um programa de rastreamento de olhar usando *Compressive Sensing* e concluímos que esse rastreador apresenta resultado semelhante a rastreadores comerciais.



# Capítulo 2

## Rastreamento de olhar

Rastreamento de olhar consiste em determinar a direção do olhar, ou seja, a partir da posição do olho identificar para onde uma pessoa está olhando em determinado instante e também mudanças na direção do olhar em determinado período [14].

Abaixo listamos algumas aplicações de rastreamento de olhar.

### 2.1 Aplicações

Podemos usar o rastreamento de olhar para construir interfaces para ajudar pessoas com dificuldades motoras a interagir com computadores [12]. Kurauchi et al.[10] desenvolveram um teclado virtual que permite a digitação através do olhar, como mostra a Figura 2.1.

Através da análise da posição do olhar é possível avaliar a influência de um anúncio sobre a atenção dos consumidores e, assim, ajudar na criação de propagandas mais eficientes [5]. A Figura 2.2 mostra as regiões onde as pessoas mais prestam atenção ao observar um anúncio.

O olhar também pode ser usado como um indicador de usabilidade de interfaces de *software*, sendo usado em estudos de interação humano-computador (IHC) [5]. Um estudo feito por [15] mostra que usuários tendem a ignorar informações que parecem propagandas em um site. No caso, solicitaram para os participantes encontrarem a população dos Estados





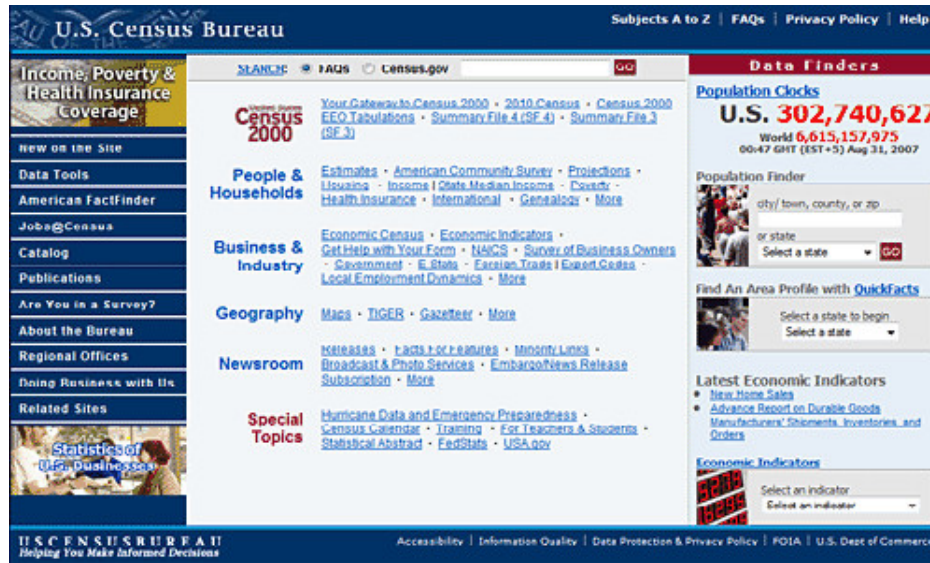
Figura 2.2: O rastreamento de olhar pode ser usado para avaliar a eficiência de uma propaganda. Regiões coloridas indicam onde as pessoas prestam mais atenção no anúncio. Reproduzido de <http://www.businessinsider.com.au/eye-tracking-heatmaps-2014-7>

1. **Eletro-oculografia**, que registra diferenças de potencial elétrico na pele ao redor da cavidade ocular.
2. **Lente de contato**, uma bobina é instalada no olho sobre uma lente de contato e a posição do olho é estimada ao medir o campo eletromagnético. [4]. Este método é o mais preciso para medir movimentos do olho, porém é o mais desconfortável [4].
3. **Rastreamento baseado em vídeo**, que usa técnicas de processamento de imagens para identificar a posição da pupila nas imagens do olho. Esse método é mais confortável que os demais métodos e razoavelmente preciso.

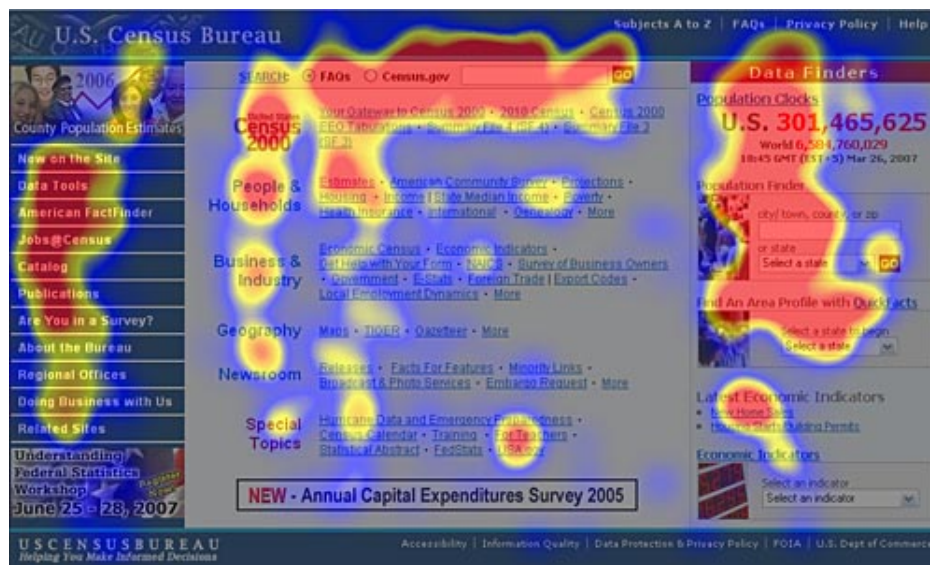
Dois tipos de imagem são comumente usados no rastreamento baseado em vídeo: imagens no espectro visual ou imagens em infravermelho.

No rastreamento baseado em vídeo, a luz refletida pelos olhos é registrada. Neste caso, a eficiência do rastreamento depende das condições de iluminação do ambiente, o que torna o processo complicado [12].

No caso de imagens em infravermelho, não temos esse problema, pois o olho é constantemente iluminado por uma fonte de luz infravermelha, que não é percebida pelo usuário.



(a)



(b)

Figura 2.3: **a** Imagem original do site. Regiões vermelhas em **(b)** são as mais observadas no site por participantes procurando a população dos Estados Unidos. Note que a região com essa informação, no canto superior direito, é parcipalmente observada, indicando que foi ignorada pelos usuários. Reproduzido de [15].

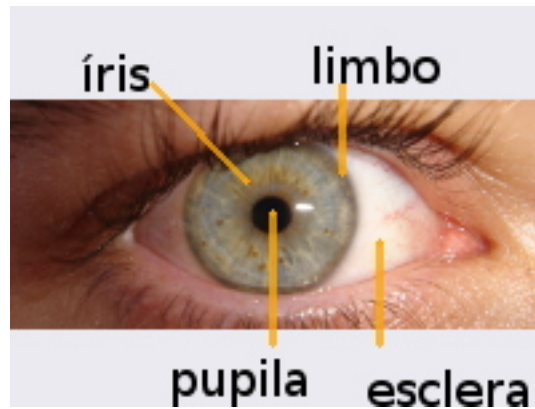


Figura 2.4: Regiões do olho. Adaptado de [http://commons.wikimedia.org/wiki/File:My\\_eye.jpg](http://commons.wikimedia.org/wiki/File:My_eye.jpg)

Uma vantagem desta abordagem é que a pupila reflete parte da luz infravermelha recebida, sendo a região mais brilhante do olho na imagem [12]. Devido ao seu tamanho e posição, a pupila tem menor chance de ser parcialmente ocultada pelos cílios do que a esclera, outra região que reflete o infravermelho.

A principal desvantagem é que não é possível usar esse tipo de rastreador ao ar livre durante o dia devido à luz infravermelha do ambiente [12].

As técnicas de rastreamento de olhar também variam de acordo com a localização da câmera, que pode ser instalada junto à cabeça do usuário (*head-mounted*) ou remotamente. No caso do rastreamento *head-mounted*, o usuário deve usar um acessório equipado com uma câmera que registra as imagens do olho. A figura 2.5 mostra um rastreador desse tipo

Uma vantagem do rastreamento *head-mounted* é que a câmera se move com a cabeça, então mudanças de pose não causam mudanças da posição da pupila na imagem. A desvantagem é a necessidade de usar um equipamento acoplado à cabeça. Por estar muito próxima do olho, a câmera do olho pode ocultar parte do campo de visão, dificultando a interação com o usuário.



Figura 2.5: Um rastreador de olhar acoplado à cabeça. Reproduzido de <https://pupil-labs.com/blog/2014-01/new-pupil-pro-headset-capture-software-0-3-7/>

## 2.3 Características relevantes

Em algumas aplicações de rastreamento de olhar não estamos interessados apenas na direção do olhar em determinado instante, mas também podemos querer observar outras características relacionadas ao movimento dos olhos. As características mais relevantes são [14]:

- **Fixação e sacada:** Em 1879, Emile Java (oftalmologista francês) observou que os movimentos do olho não ocorrem de forma contínua, e sim de movimentos rápidos, chamados de sacadas, seguidos por breves pausas, conhecidas como fixações [14];
- **Área de interesse:** Região no ambiente que está presente no campo visual e que é de interesse em determinada pesquisa;
- **Duração do olhar:** Duração de um intervalo de tempo em que uma série de fixações está dentro de uma área de interesse;
- **Caminho de varredura:** localização espacial de uma sequência de fixações.

# Capítulo 3

## *Compressive Sensing*

*Compressive Sensing* (CS), também conhecido como *Compressive Sampling* ou *Compressed Sensing*, estuda formas de reconstruir um vetor a partir de poucas amostras. De agora em diante, iremos assumir que as amostras  $(y_i)_{i=1}^m$ , representadas pelo vetor  $y \in \mathbb{R}^m$  de um sinal  $x \in \mathbb{R}^n$  são obtidas a partir de uma transformação linear  $y = Ax$ , onde  $A$  é uma matriz  $m \times n$ , com  $m < n$ .

**Observação:** Durante o texto, sempre que referirmos a um vetor como uma imagem, assumimos que o vetor é calculado ao empilhar todas as linhas da imagem em escala de cinza.

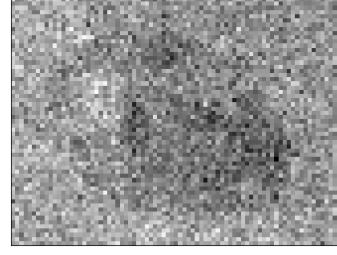
Como  $m < n$ , a equação não possui uma única solução  $x$ , se possuir. Podemos escolher o vetor  $x$  que minimiza  $\|Ax - y\|_2^2$ , pelo método de mínimos quadrados.  $x$  pode ser calculado como  $x = A^\dagger y$ , onde  $A^\dagger$  denota a pseudo-inversa de  $A$ . Porém as imagens recuperadas dessa forma não são parecidas com a imagem original.

A Figura 3.1 mostra uma imagem recuperada pelo método de mínimos quadrados para uma matriz  $A$   $m \times n$ , onde  $m = 80\% \cdot n$  (compressão de 20%) e as entradas de  $A$  são independentes e seguem a distribuição normal com média 0 e variância 1.

Sabemos que geralmente imagens são esparsas no espectro da frequência. Uma imagem pode ser decomposta como uma combinação linear de senoidais e, na prática, os coeficientes



(a) imagem original



(b) compressão de 20%

Figura 3.1: Recuperando imagem usando o método de mínimos quadrados para  $y = Ax$ , onde o vetor  $x$  representa a imagem original e  $A$  é uma matriz  $m \times n$  com  $m = 80\% \cdot n$  (compressão de 20%) cujas entradas seguem a distribuição normal de média 0 e variância 1 e são calculadas independentemente. Adaptado de [https://commons.wikimedia.org/wiki/File:5-stip\\_LHB\\_09.jpg](https://commons.wikimedia.org/wiki/File:5-stip_LHB_09.jpg)

correspondentes às frequências mais altas são muito próximos de zero. Na Figura 3.2 a imagem é reconstruída apenas com as frequências cujos coeficientes estão entre os 60% maior valor absoluto (compressão de 40%). Note que as imagens **(a)** e **(b)** são bastante parecidas, indicando que a imagem é esparsa, ou aproximadamente esparsa (com muitos coeficientes próximos de zero) no espectro da frequência.

A ideia é então escolher, entre todas as soluções possíveis, o vetor  $x$  com o maior número de coeficientes nulos. Para entender melhor o problema, as seguintes definições, retiradas de [2], serão enunciadas:

**Def:** Seja  $x \in \mathbb{R}^n$ .  $x$  é  $k$ -esparso se possui, no mínimo,  $k$  coeficientes nulos.

**Def:** Dado  $x \in \mathbb{R}^n$ , a ‘norma’ 0 de  $x$  é definida como  $\|x\|_0 = \#\{i : |x_i| > 0\}$ , ou seja, a quantidade de elementos não nulos de  $x$ .<sup>1</sup>

Observe que a ‘norma’ 0 não é uma norma, pois, se  $x = (1, 0)$ ,  $\|2x\| = 1 \neq 2\|x\|$ . Porém,  $\|\cdot\|_0$  satisfaz as propriedades demonstradas na Proposição 1, retirada de [2].

**Proposição 1.** Fixados  $x, y \in \mathbb{R}^n, \lambda \in \mathbb{R}$ , a ‘norma’ 0 satisfaz as seguintes propriedades:

---

<sup>1</sup>Denotaremos a quantidade de elementos em um conjunto  $I$  arbitrário como  $\#I$ .



$$(i) \quad \|x\|_0 \geq 0$$

$$(ii) \quad \|x\|_0 = 0 \Leftrightarrow x = 0$$

$$(iii) \quad \text{Se } \lambda \neq 0, \|\lambda x\|_0 \neq |\lambda| \|x\|_0$$

$$(iv) \quad \|x + y\|_0 \leq \|x\|_0 + \|y\|_0$$

$$(v) \quad \text{A norma } 0 \text{ não é uma norma.}$$

*Demonstração.* Fixados  $x, y \in \mathbb{R}^n, \lambda \in \mathbb{R}$

$$(i) \quad \|x\|_0 = \#\{i \in \{1, \dots, n\} : |x_i| > 0\} \geq 0$$

$$(ii)$$

$$\begin{aligned} \|x\|_0 = 0 &\Rightarrow \#\{i \in \{1, \dots, n\} : |x_i| > 0\} = 0 \\ &\Rightarrow \forall i \in \{1, \dots, n\}, |x_i| = 0 \\ &\Rightarrow x = 0 \end{aligned}$$

$$(iii) \quad \text{Se } \lambda \neq 0,$$

$$\begin{aligned} \|\lambda x\|_0 &= \#\{i \in \{1, \dots, n\} : |\lambda x_i| > 0\} \\ &= \#\{i \in \{1, \dots, n\} : |x_i| > 0\} \\ &= \|x\|_0 \end{aligned}$$

Então, para  $x \neq 0$  e  $\lambda \notin \{0, 1\}$ , temos que

$$\begin{aligned} \|\lambda x\|_0 &= \|x\|_0, \text{ pelo item (i)} \\ &\neq |\lambda| \|x\|_0 \end{aligned}$$



(a) imagem original



(b) compressão de 40%

Figura 3.2: A imagem em **(b)** é calculada ao descartar 40% das senoidais que compõem a imagem cujos coeficientes têm menor módulo. Adaptado de [https://commons.wikimedia.org/wiki/File:Along\\_The\\_Main\\_Ride\\_-\\_Beale\\_Arboretum\\_-\\_West\\_Lodge\\_Park\\_-\\_Hadley\\_Wood\\_-\\_Enfield\\_London\\_2.jpg](https://commons.wikimedia.org/wiki/File:Along_The_Main_Ride_-_Beale_Arboretum_-_West_Lodge_Park_-_Hadley_Wood_-_Enfield_London_2.jpg)

(iv)

$$\begin{aligned}\|x\|_0 + \|y\|_0 &= \#\{i : |x_i| > 0\} + \#\{i : |y_i| > 0\} \\ &\geq \#(\{i : |x_i| > 0\} \cup \{i : |y_i| > 0\}) \\ &= \#\{i : |x_i| > 0 \text{ ou } |y_i| > 0\} \\ &= \#\{i : |x_i| + |y_i| > 0\} \\ &\geq \#\{i : |x_i + y_i| > 0\} \\ &= \|x + y\|_0\end{aligned}$$

(v) Sejam  $\lambda = 2, x = e_1$ , então

$$\|\lambda x\|_0 = \|x\|_0 = 1 \neq |\lambda| \|x\|_0$$

Então a norma 0 não é uma norma.

□

O problema que resolveremos para encontrar o vetor  $x$  mais esparsamente amostrado como  $x = Ay$  será formulado do seguinte modo:

$$\min_{x \in \mathbb{R}^n} \|x\|_0 \text{ sujeito a } Ax = y \quad (P_0)$$

No entanto,  $(P_0)$  é NP-difícil [7] (NP, ou tempo polinomial não determinístico, denota a classe de problemas que podem ser verificados em tempo polinomial. Ainda não sabemos se esses problemas podem ser resolvidos em tempo polinomial). Então tentamos resolver o seguinte problema:

$$\min_{x \in \mathbb{R}^n} \|x\|_1 \text{ sujeito a } Ax = y \quad (P_1)$$

A Figura 3.3 nos ajuda a entender a semelhança entre  $(P_0)$  e  $(P_1)$ . A figura mostra o problema de minimização de  $\|x\|_p$  sujeito a  $Ax = y$  para  $x \in \mathbb{R}^2$  e  $y \in \mathbb{R}$ . O ponto  $x$  minimiza

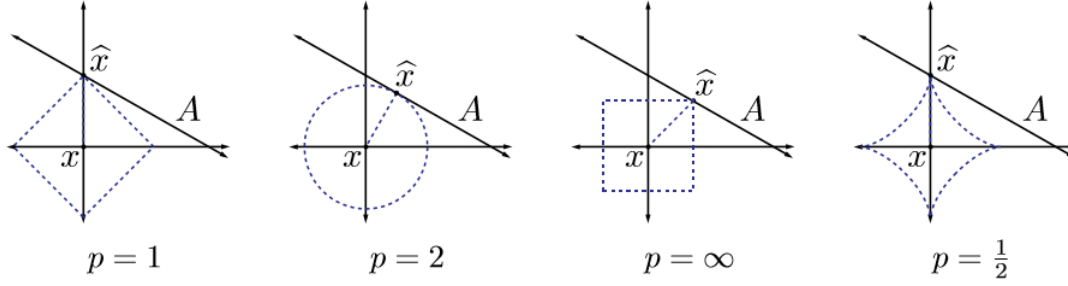


Figura 3.3: Pontos na reta  $Ax = y$  que minimizam a norma  $\|\cdot\|_p$  para diferentes valores de  $p$ . As soluções encontradas para  $p = 1$  e  $p = 1/2$  são esparsas. Note que  $\|\cdot\|_{\frac{1}{2}}$  não é norma. Reproduzido de [2].

a norma em um único ponto se a fronteira de uma bola fechada centrada em 0 toca a reta  $Ax = y$  apenas em  $x$ . Neste caso, as soluções calculadas para  $p = 1$  e  $p = 1/2$  são esparsas. Note que  $\|\cdot\|_{\frac{1}{2}}$  não é norma.

CS tem como objetivo estudar condições para garantir a equivalência entre os dois problemas, e condições para a unicidade de soluções de  $(P_0)$  e  $(P_1)$ . Descrevemos estas condições nas seções 3.1, 3.2 e 3.4.3. Todos resultados mostrados nessas seções podem ser encontrados em [2], exceto quando indicados explicitamente.

### 3.1 Unicidade de $(P_0)$

Se a matriz  $A$  fosse  $n \times n$  de posto máximo, a unicidade estaria garantida. Como  $m < n$ , definimos o conceito de *spark*, que verbalmente é uma fusão dos conceitos ‘esparso’ e ‘posto’ (*rank*) [11].

**Definição 1.** *Seja  $A$  uma matriz  $m \times n$ . O *spark* de  $A$ , denotado por  $\text{spark}(A)$ , é o menor número de colunas linearmente dependentes de  $A$  (ou de forma equivalente,  $\text{spark}(A)$  é o menor número de colunas LD de  $A$ )<sup>2</sup>.*

---

<sup>2</sup>Dizemos que um conjunto de colunas é LD se elas forem linearmente dependentes. Diremos que uma matriz  $A$  é LD se suas colunas forem LD

O Teorema a seguir nos indica uma condição que  $A$  deve satisfazer para garantir a unicidade de  $(P_0)$

**Teorema 1.** *Seja  $A$  uma matriz  $m \times n$ . São equivalentes:*

$$(i) \text{ spark}(A) > 2k$$

$$(ii) \text{ Para todo } y \in \mathbb{R}^m, \text{ existe no máximo um } x \in \Sigma_k \text{ tal que } y = Ax.$$

Portanto, é desejável que  $\text{spark}(A)$  seja alto pois, nesse caso, seria possível recuperar vetores  $x$  não tão esparsos.

Antes de demonstrar o teorema, enunciaremos uma definição e demonstraremos um lema.

**Definição 2.** *Dado  $k \in \{1, \dots, n\}$ ,  $\Sigma_k = \{x : \|x\|_0 \leq k\} = \{x : x \text{ é } k\text{-esparso}\}$*

**Lema 1.** *Dados  $u, v \in \Sigma_k, u - v \in \Sigma_{2k}$*

*Demonstração.* Sejam  $u, v \in \Sigma_k$ . Então

$$\begin{aligned} \|u - v\|_0 &\leq \|u\|_0 + \|v\|_0 \\ &= \|u\|_0 + \|v\|_0 \\ &\leq k + k = 2k \\ &\Rightarrow u - v \in \Sigma_{2k} \end{aligned}$$

□

Agora iremos demonstrar o Teorema 1.

*Demonstração.* Seja  $A$  uma matriz  $m \times n$ . Denote cada coluna  $j$  de  $A$  por  $a_j$ .

$$\bullet \neg(i) \Rightarrow \neg(ii)$$

Suponha que  $\text{spark}(A) \leq 2k$ . Então existe um conjunto  $J \subset \{1, \dots, n\}$  tal que  $\#J = 2k$  e existe uma bijeção  $\mu$  entre  $\{1, \dots, 2k\}$  e  $J$  tais que

$\{a_{\mu(j)} : j \in \{1, \dots, 2k\}\}$  é LD

Então existe  $c \in \mathbb{R}^{2k}, c \neq 0$  tal que

$$\sum_{j=1}^{2k} c_j a_{\mu(j)} = 0$$

Defina  $u, v \in \mathbb{R}^n$  tais que, para  $j \in \{1, \dots, n\}$

$$u_j = \begin{cases} c_{\mu^{-1}(j)} & \text{se } j \in \mu^{-1}(\{1, \dots, k\}) \\ 0, & \text{caso contrário} \end{cases}$$

$$v_j = \begin{cases} -c_{\mu^{-1}(j)} & \text{se } j \in \mu^{-1}(\{k+1, \dots, 2k\}) \\ 0 & \text{caso contrário} \end{cases}$$

Se  $j \in \mu^{-1}(\{1, \dots, 2k\})$ ,  $u_j - v_j = c_j$ . Então  $u \neq v$

Então  $\|u\|_0, \|v\|_0 \leq k \Rightarrow u, v \in \Sigma_k$ .

Então

$$\begin{aligned} \sum_{j=1}^{2k} c_j a_{\mu(j)} &= \sum_{j=1}^k c_j a_{\mu(j)} + \sum_{j=k+1}^{2k} c_j a_{\mu(j)} \\ &= \sum_{j=1}^k u_{\mu(j)} a_{\mu(j)} - \sum_{j=k+1}^{2k} v_{\mu(j)} a_{\mu(j)} \\ &= Au - Av = 0 \\ &\Rightarrow Au = Av \end{aligned}$$

Então, definindo  $y = Au$ , existem  $u, v \in \mathbb{R}^n$  distintos tais que  $Au = Av$ .

- $\neg(\text{ii}) \Rightarrow \neg(\text{i})$

Suponha que existam  $y \in \mathbb{R}^m$ ,  $z, w \in \Sigma_k$  distintos tais que  $y = Az = Aw$ .

Seja  $x = z - w$ . Então  $Ax = 0$  e, pelo Lema 3.1,  $x \in \Sigma_{2k}$ . Além disso,  $x \neq 0$ .

Defina  $K = \{i \in \{1, \dots, n\} : |x_i| > 0\}$  e  $\mu$  uma bijeção entre  $\{1, \dots, \|x\|_0\}$  e  $K$ . Então

$$Ax = \sum_{j=1}^{\|x\|_0} a_{\mu(j)} x_{\mu(j)} = 0$$

Como  $x \neq 0$ , existe algum  $j \in \{1, \dots, \|x\|_0\}$  tal que  $x_{\mu(j)} \neq 0$ . Logo,  $A$  possui  $\|x\|_0$  colunas linearmente dependentes. Portanto,  $\text{spark}(A) \leq \|x\|_0 \leq 2k$ .

□

## 3.2 Unicidade de $(P_1)$

Ao encontrar a solução única de  $(P_0)$ , encontramos o vetor desejado. No entanto,  $(P_0)$  é NP-difícil. Uma alternativa seria resolver o problema  $(P_1)$ .

Precisamos então estudar a unicidade de soluções para  $(P_1)$  e condições para a equivalência entre  $(P_0)$  e  $(P_1)$ , ou seja, condições em que um vetor  $x$  é solução única de ambos  $(P_0)$  e  $(P_1)$ .

Definimos o conceito de *Null space property* para estudar a unicidade de  $(P_1)$ .

**Definição 3.** Seja  $\Lambda \subset \{1, 2, \dots, n\}$ . Defina  $\Lambda^c = \{1, 2, \dots, n\} \setminus \Lambda$ . Dado  $x \in \mathbb{R}^n$ ,  $x_\Lambda \in \mathbb{R}^n$ , onde, para  $i \in [n] = \{1, \dots, n\}$

$$(x_\Lambda)_i = \begin{cases} x_i, & \text{se } i \in \Lambda \\ 0, & \text{caso contrário} \end{cases}$$

**Definição 4.** Uma matriz  $A$  de tamanho  $m \times n$  satisfaz a *null space property* (NSP) relativa ao conjunto  $\Lambda \subset [n]$  se para todo  $h \in \ker(A) \setminus \{0\}$

$$\|h_\Lambda\|_1 < \|h_{\Lambda^c}\|_1$$

**Definição 5.** Dizemos que uma matriz de tamanho  $m \times n$  satisfaz a null space property (NSP) de ordem  $k$  se ela satisfaz a NSP para todo  $\Lambda \subset [n]$  com  $\#\Lambda \leq k$ .

**Lema 2.** Seja  $\Lambda \in [n]$ . Dado  $x \in \mathbb{R}^n$ ,

$$\|x\|_1 = \|x_\Lambda\|_1 + \|x_{\Lambda^c}\|_1$$

*Demonstração.*

$$\|x\|_1 = \sum_{j=1}^n |x_j| = \sum_{j \in \Lambda} |x_j| + \sum_{j \in \Lambda^c} |x_j| = \|x_\Lambda\|_1 + \|x_{\Lambda^c}\|_1$$

□

**Teorema 2.** Seja  $A$  uma matriz  $m \times n$ . Todo vetor  $x$  de suporte  $\Lambda$  tal que  $Ax = y$  é solução única de  $(P_1)$  se e somente se  $A$  satisfaz a NSP para  $\Lambda$ .

*Demonstração.* Seja  $A$  uma matriz  $m \times n$ .

•  $(\Rightarrow)$

Seja  $h \in \ker(A) \setminus \{0\}$ . Então

$$0 = Ah = Ah_\Lambda + Ah_{\Lambda^c} \Rightarrow Ah_\Lambda = -Ah_{\Lambda^c} = A(-h_{\Lambda^c})$$

Por hipótese,  $h_\Lambda$  é solução única de  $(P_1)$  com  $y = Ah_\Lambda$ , pois tem suporte  $\Lambda$ . Como  $Ah_\Lambda = A(-h_{\Lambda^c})$ , temos que

$$\|h_\Lambda\|_1 < \|-h_{\Lambda^c}\|_1 = \|h_{\Lambda^c}\|_1$$

Portanto  $A$  satisfaz NSP para  $\Lambda$ .



- ( $\Leftarrow$ )

Seja  $x$  de suporte  $\Lambda$  com  $Ax = y$ . Suponha que  $A$  satisfaz NSP para  $\Lambda$ .

Seja  $z \neq x$  tal que  $Az = Ax$ . Então  $z - x \in \ker A \setminus \{0\}$ . Como  $A$  satisfaz NSP para  $\Lambda$ , temos que

$$\begin{aligned} \|z_\Lambda - x_\Lambda\|_1 &< \|z_{\Lambda^c} - x_{\Lambda^c}\|_1 = \|z_{\Lambda^c}\|_1 \\ \Rightarrow \|z_\Lambda - x_\Lambda\|_1 + \|z_\Lambda\|_1 &< \|z_{\Lambda^c}\|_1 + \|z_\Lambda\|_1 \\ \Rightarrow \|z_\Lambda - x_\Lambda\|_1 + \|z_\Lambda\|_1 &< \|z\|_1 \\ \Rightarrow \|x\|_1 &< \|z\|_1, \text{ pela desigualdade triangular.} \end{aligned}$$

Portanto  $x$  é solução única de  $(P_1)$ .

□

A noção de NSP nos diz que a norma 1 dos vetores do núcleo de  $A$  não estão concentradas num número pequeno de índices. O Corolário 1 estabelece uma relação entre NSP e a unicidade de soluções de  $(P_1)$ .

**Colorário 1.** *Seja  $A$  uma matriz  $m \times n$ . Todo vetor  $x$   $k$ -esparso tal que  $Ax = y$  é solução única de  $(P_1)$  se e somente se  $A$  satisfaz a NSP de ordem  $k$ .*

*Demonstração.* Seja  $A$  uma matriz  $m \times n$ .

- ( $\Rightarrow$ )

Seja  $\Lambda \subset [n]$  tal que  $\#\Lambda = k$ . Seja  $x$  um vetor de suporte em  $\Lambda$  tal que  $x$  é solução única de  $(P_1)$ . Então, pelo Teorema 2,  $A$  satisfaz NSP para  $\Lambda$ . Como  $\Lambda$  é arbitrário,  $A$  satisfaz NSP de ordem  $k$ .

- ( $\Leftarrow$ )

Seja  $x$  um vetor  $k$ -esparso tal que  $Ax = y$ . Então existe  $\Lambda \subset [n]$  tal que  $\#\Lambda = k$  e  $x_\Lambda = x$ . Como  $A$  satisfaz NSP para  $\Lambda$ ,  $x$  é solução única.

□

### 3.3 Equivalência entre $(P_0)$ e $(P_1)$

Vamos demonstrar a equivalência entre  $(P_0)$  e  $(P_1)$  para  $A$  com colunas  $a_i$  de norma  $\|a_i\|_2 = 1$ . Definimos também o conceito de coerência.

**Definição 6.** *Seja  $A$  uma matriz  $m \times n$  cujas colunas  $a_i$  possuem  $\|a_i\|_2 = 1$ . definimos a coerência (ou mutual coherence) de  $A$ , denotada por  $\mu(A)$ , como*

$$\mu(A) = \max_{i \neq j} |\langle a_i, a_j \rangle|$$

O Teorema 3 garante a equivalência entre  $(P_0)$  e  $(P_1)$ .

**Teorema 3.** *Seja  $x$  uma solução de  $(P_0)$  tal que*

$$\|x\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(A)} \right)$$

*Então  $x$  é solução única de  $(P_0)$  e  $(P_1)$ .*

Alguns resultados serão apresentados antes de demonstrar o Teorema 3.

**Definição 7.** *Seja  $x \in \mathbb{R}^n$ . Definimos  $|x|$  como um vetor  $|x| \in \mathbb{R}^n$  tal que para cada  $i \in [n]$ ,  $|x|_i = |x_i|$ .*

**Definição 8.** *Dados  $x, y \in \mathbb{R}^n$ ,  $x \leq y$  se  $\forall j \in [n]$ ,  $x_j \leq y_j$ .*

**Lema 3.** *Fixado  $x \in \mathbb{R}^n$ ,  $x \leq |x|$ .*

*Demonstração.*  $\forall i \in [n]$ ,  $x_i \leq |x_i| = |x|_i$

□

**Lema 4.** *Dado  $x \in \mathbb{R}^n$ ,  $\|x\|_1 = \||x|\|_1$*

*Demonstração.* Seja  $x \in \mathbb{R}^n$ ,

$$\|x\|_1 = \sum_{i=1}^n |x_i| = \sum_{i=1}^n |(|x_i|)| = \sum_{i=1}^n |(|x|_i)| = \| |x| \|_1$$

□

**Definição 9.** *Seja  $A$  uma matriz  $m \times n$ .  $|A|$  é uma matriz  $m \times n$  cujas entradas  $|A|_{ij} = |A_{ij}|$ , para todo  $i \in [m], j \in [n]$ .*

**Lema 5.** *Sejam  $A$  uma matriz  $m \times n$  e  $x \in \mathbb{R}^n$ . Então  $|Ax| \leq |A||x|$ .*

*Demonstração.* Dado  $i \in [n]$ ,

$$|Ax|_i = |(Ax)_i| = \left| \sum_{k=1}^n A_{ik}x_k \right| \leq \sum_{k=1}^n |A_{ik}x_k| = \sum_{k=1}^n |A_{ik}||x_k| = (|A||x|)_i$$

□

**Teorema 4.** (Geršgorin) [9] *Seja  $A$  uma matriz  $n \times n$ , de elementos  $a_{ij}, 1 \leq i, j \leq n$ . Então, para cada autovalor  $\lambda$  de  $A$ ,*

$$\lambda \in \bigcup_{i=1}^n \overline{B}(a_{ii}, r_i)$$

onde

$$r_i = \sum_{j \neq i} |a_{ij}|$$

*Demonstração.* Sejam  $\lambda \in \mathbb{C}$  um autovalor de  $A$ ,  $v \in \mathbb{C}^n \setminus \{0\}$  tal que  $Av = \lambda v$ . Escolha  $i \in [n]$  de forma que  $|v_j| \leq |v_i|$ , para todo  $j \in [n]$ . Então  $v_i \neq 0$  e

$$\begin{aligned}
\lambda v_i &= (Av)_i = \sum_{j=1}^n a_{ij}v_j = a_{ii}v_i + \sum_{j \neq i} a_{ij}v_j \\
\Rightarrow (\lambda - a_{ii})v_i &= \sum_{j \neq i} a_{ij}v_j \\
\Rightarrow |\lambda - a_{ii}||v_i| &\leq \sum_{j \neq i} |a_{ij}||v_j| \\
\Rightarrow |\lambda - a_{ii}| &\leq \sum_{j \neq i} |a_{ij}| \frac{|v_j|}{|v_i|} \leq \sum_{j \neq i} |a_{ij}| = r_i \\
\Rightarrow \lambda &\in \overline{B}(a_{ii}, r_i)
\end{aligned}$$

□

**Lema 6.** Para toda matriz real  $A$  de tamanho  $m \times n$ , com colunas  $a_i$  tais que  $\|a_i\|_2 = 1$ ,

$$\text{spark}(A) \geq 1 + \frac{1}{\mu(A)}$$

*Demonstração.* Sejam  $p = \text{spark}(A)$  e  $\Lambda \subset [n]$  tal que  $\#\Lambda = p$ .

Seja  $A_\Lambda$  uma matriz  $m \times p$  onde, para cada  $i \in \Lambda$ ,  $a_i$  é coluna de  $A_\Lambda$ . Então  $A_\Lambda$  é LD. Então existe  $v \in \mathbb{R}^n \setminus \{0\}$  tal que

$$\begin{aligned}
A_\Lambda v &= 0 \\
\Rightarrow v^T A_\Lambda^T A_\Lambda v &= 0 \Rightarrow v^T G v = 0, \text{ definindo } G = A_\Lambda^T A_\Lambda
\end{aligned}$$

Então 0 é um autovalor de  $G$ .

Denotaremos os elementos de  $G$  por  $g_{ij}$ . Note que  $g_{ij} = \langle a_i, a_j \rangle$  e que  $g_{ii} = 1$ .

Pelo Teorema de Geršgorin), existe  $i \in [n]$  tal que

$$\begin{aligned}
g_{ii} &\leq \sum_{i \neq j} |g_{ij}| \\
\Rightarrow 1 &\leq \sum_{i \neq j} |\langle (A_\Lambda)_i, (A_\Lambda)_j \rangle| \\
&\leq (p-1)\mu(A)
\end{aligned}$$

Logo,

$$p = \text{spark}(A) \geq 1 + \frac{1}{\mu(A)}$$

□

Agora demonstraremos o Teorema 3.

*Demonstração.* Assumindo que  $x$  é solução única de  $(P_0)$

Seja  $z \in \mathbb{R}^n$  tal que  $z \neq x$ . Então  $z = x + h$  para algum  $h \in \ker(A) \setminus \{0\}$ . Seja  $\Lambda$  o suporte de  $x$ . Temos que

$$\begin{aligned}
\|z\|_1 - \|x\|_1 &= \sum_{k=1}^n |x_k + h_k| - \sum_{k=1}^n |x_k| \\
&= \sum_{k \in \Lambda} |x_k + h_k| + \sum_{k \in \Lambda^c} |x_k + h_k| - \sum_{k \in \Lambda} |x_k| \\
&= \sum_{k \in \Lambda} |x_k + h_k| + \sum_{k \in \Lambda^c} |h_k| - \sum_{k \in \Lambda} |x_k| \\
&= \sum_{k \in \Lambda^c} |h_k| + \sum_{k \in \Lambda} (|x_k + h_k| - |x_k|) \\
&\geq \sum_{k \in \Lambda^c} |h_k| - \sum_{k \in \Lambda} |h_k|, \text{ pois cada } |x_k + h_k| - |x_k| \geq -|h_k| \\
&= \|h_{\Lambda^c}\|_1 - \|h_\Lambda\|_1 \\
&= \|h_{\Lambda^c}\|_1 - \|h_\Lambda\|_1 + (\|h_\Lambda\|_1 - \|h_\Lambda\|_1) \\
&= \|h\|_1 - 2\|h_{\Lambda^c}\|_1
\end{aligned}$$

Como  $\forall h, Ah = 0 \Rightarrow A^T Ah = 0$ , temos que

$$\inf_{\substack{Ah=0 \\ \|h\|_1=1}} 1 - 2\|h_\Lambda\|_1 \geq \inf_{\substack{A^T Ah=0 \\ \|h\|_1=1}} 1 - 2\|h_\Lambda\|_1 \quad (*)$$

Como  $\|h\|_1 = \||h|\|_1$  e  $|h_\Lambda| = |h|_\Lambda$ , temos que

$$\inf_{\substack{A^T Ah=0 \\ \|h\|_1=1}} 1 - 2\|h_\Lambda\|_1 \geq \inf_{\substack{A^T Az=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2\|z_\Lambda\|_1 \quad (**)$$

Definindo  $G = A^T A$ , cada entrada  $G_{ij} = \langle a_i, a_j \rangle$ .

$$\inf_{\substack{A^T Az=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2\|z_\Lambda\|_1 = \inf_{\substack{Gz=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2\|z_\Lambda\|_1$$

Como  $z \geq 0$  e  $Gz = 0$ ,

$$z = |-z| \Rightarrow z = |Gz - z| = |(G - I)z| \leq |G - I|z$$

Todos as entradas de  $G - I$  maiores os iguais a zero e os elementos da diagonal são nulos, então

$$|G - I|_{ij} \leq \max_{i \neq j} G_{ij} = \max_{i \neq j} \langle a_i, a_j \rangle = \mu(A)$$

Então

$$|G - I| \leq \mu(A)(\mathbb{1} - I)$$

Logo

$$z \leq \mu(A)(\mathbb{1} - I)z$$

Então, dado  $i \in \Lambda$ ,

$$\begin{aligned}
z_i &\leq \mu(A) \left( \sum_{j=1}^n z_j - z_i \right) \\
&\leq \mu(A) (\|z\|_1 - z_i) \\
&= \mu(A) (1 - z_i) \\
\Rightarrow z_i + z_i \mu(A) &\leq \mu(A) \\
\Rightarrow z_i &\leq \frac{\mu(A)}{1 + \mu(A)}
\end{aligned}$$

Então

$$\|z_\Lambda\|_1 = \sum_{i \in \Lambda} z_i \leq \sum_{i \in \Lambda} \frac{\mu(A)}{1 + \mu(A)} = \|x\|_0 \frac{\mu(A)}{1 + \mu(A)}$$

Então

$$\begin{aligned}
\inf_{\substack{Gz=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2\|z_\Lambda\|_1 &\geq \inf_{\substack{Gz=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2\|x\|_0 \frac{\mu(A)}{1 + \mu(A)} \\
&> \inf_{\substack{Gz=0 \\ \|z\|_1=1 \\ z \geq 0}} 1 - 2 \frac{1}{2} \left( 1 + \frac{1}{\mu(A)} \right) \frac{\mu(A)}{1 + \mu(A)} \\
&= 1 - 2 \frac{1}{2} \left( 1 + \frac{1}{\mu(A)} \right) \frac{\mu(A)}{1 + \mu(A)} \\
&= 1 - \frac{\mu(A) + 1}{\mu(A)} \frac{\mu(A)}{1 + \mu(A)} \\
&= 1 - 1 = 0
\end{aligned}$$

De (\*) e (\*\*), concluímos que

$$\inf_{\substack{Ah=0 \\ \|h\|_1=1}} 1 - 2\|h_\Lambda\|_1 > 0$$

Então, para qualquer  $h \in \ker(A) \setminus \{0\}$ , temos que

$$\left\| \frac{h}{\|h\|_1} \right\|_1 = 1 \text{ e } A \left( \frac{h}{\|h\|_1} \right) = 0$$

Então

$$\begin{aligned} 1 - 2 \left\| \left( \frac{h}{\|h\|_1} \right)_\Lambda \right\|_1 &= 1 - 2 \left\| \frac{h_\Lambda}{\|h\|_1} \right\|_1 \\ &= \frac{\|h\|_1}{\|h\|_1} - 2 \left\| \frac{h_\Lambda}{\|h\|_1} \right\|_1 > 0 \\ &\Rightarrow \|h\|_1 - 2\|h_\Lambda\|_1 > 0 \end{aligned}$$

Como  $h \in \ker(A) \setminus \{0\}$  é arbitrário, temos que para algum  $h \in \ker(A) \setminus \{0\}$ ,  $z = x + h$ .

Então

$$\|z\|_1 - \|x\|_1 = \|h\|_1 - 2\|h_{\Lambda^c}\|_1 > 0$$

Portanto  $x$  é solução única de  $(P_1)$ .

□

### 3.4 Matrizes que satisfazem as condições de CS

Como vimos anteriormente, a matriz  $A$  deve satisfazer algumas condições para garantir a equivalência entre  $(P_0)$  e  $(P_1)$ , que chamaremos de condições de CS. Em vez de construir uma matriz  $A$  de forma determinística, o que talvez não seja fácil, podemos construir uma matriz aleatória que satisfaz as condições de CS.

Uma matriz  $A$  cujas entradas são amostradas através de uma distribuição normal satisfazem as condições de CS com alta probabilidade, como mostra o Teorema 5, que foi extraído de [19].



**Teorema 5.** [19] *Sejam  $m < n$ ,  $\Omega \subset \mathbb{R}^n$  e  $A \in \mathbb{R}^{m \times n}$  tais que apenas uma das afirmações é verdadeira:*

1. *os elementos  $a_{ij}$  de  $A$  são independentes e têm distribuição normal com média 0 e variância 1.*
2.  *$A$  é uma matriz de posto  $m$  com  $BA^T = 0$  para uma matriz  $B \in \mathbb{R}^{(n-m) \times n}$  cujas entradas são independentes e têm distribuição normal com média 0 e variância 1.*

*Então com probabilidade maior que  $1 - e^{-c_0(n-m)}$ ,  $\bar{x}$  é solução única de  $(P_0)$  e  $(P_1)$  se  $\bar{x}$  satisfaz*

$$\|\bar{x}\|_0 < \frac{c_1^2}{4} \frac{m}{1 + \log(n/m)}$$

*onde  $c_0, c_1 > 0$  são constantes independentes das dimensões  $m$  e  $n$ .*

# Capítulo 4

## O algoritmo de homotopia

O algoritmo de Homotopia [3] é um algoritmo com o objetivo de resolver  $(P_1)$  minimizando funções do tipo

$$J_{\lambda_n}(x) = \frac{1}{2}\|Ax - y\|_2^2 + \lambda_n\|x\|_1$$

para uma sequência decrescente  $(\lambda_n)$  de números positivos.

Podemos encontrar o mínimo de uma função diferencial  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  procurando os pontos  $x$  no domínio em que  $\nabla f(x) = 0$ . No nosso caso, a norma 1 não é diferenciável. Então definimos uma generalização para a diferencial, que será usada no algoritmo, chamada de Homotopia.

**Definição 10.** *Seja  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . A subdiferencial de  $f$  em  $x \in \mathbb{R}$  é dada por*

$$\partial f(x) = \{v \in \mathbb{R}^N : f(y) - f(x) \geq \langle v, y - x \rangle, \forall y \in \mathbb{R}^N\}$$

Não é difícil ver que o operador  $\partial$  é linear. A subdiferencial coincide com a diferencial de uma função, quando ela for convexa e diferenciável, como mostra a seguinte proposição [1], que vamos assumir sem demonstração:

**Proposição 2.** *Se  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  for convexa e diferenciável, então para todo  $x \in \mathbb{R}^n$ ,  $\partial f(x) = \{\nabla f(x)\}$ .*

Um ponto  $x_0$  é ponto de mínimo de uma função  $f$  se, e somente se,  $0 \in \partial f$ , pois

$$\forall x \in \mathbb{R}^n, f(x) - f(x_0) \geq 0 = \langle 0, x - x_0 \rangle$$

Como a norma 1 é convexa, sua subdiferencial em um ponto  $x$  é um vetor  $v$  tal que para cada  $i = 1, \dots, n$ ,

$$v_i = \begin{cases} \text{sgn}(v_i) & \text{se } v_i \neq 0 \\ \{1, -1\} & \text{se } v_i = 0 \end{cases}$$

O algoritmo de Homotopia [6] [3] minimiza a função em um número finito de passos. Fixado um  $\lambda$  positivo, considere a seguinte função :

$$J_\lambda(x) = \frac{1}{2} \|Ax - y\|_2^2 + \lambda \|x\|_1$$

e seu respectivo minimizador  $x_\lambda$ . Quando  $\lambda$  é grande,  $x_\lambda = 0$ . O ponto  $\tilde{x}$ , solução de  $(P_1)$  é solução de algum  $J_{\tilde{\lambda}}$ . A curva  $\lambda \mapsto x_\lambda$  é linear por partes, então é possível encontrar a solução de  $(P_1)$  com um número finito de passos.

A subdiferencial de  $J_\lambda$  é

$$\partial J_\lambda(x) = A^T(Ax - y) + \lambda \partial \|x\|_1$$

Se  $x = x_\lambda$  então  $0 \in \partial J_\lambda(x)$  é equivalente a

$$\begin{cases} (A^*(Ax - y))_i = \lambda \text{sgn}(x_i), & \text{se } x_i \neq 0 \\ |A^*(Ax - y)|_i \leq \lambda, & \text{se } x_i = 0 \end{cases} \quad (4.1)$$

para  $i \in \{1, \dots, n\}$ , onde  $\text{sgn}(x_i) = 1$  se  $x_i \geq 0$  e 0 caso contrário.

Escrevendo  $c = A^T(Ax - y)$  e denotando  $I$  como o suporte de  $x$ , então (4.1) equivale a

$$\begin{cases} c(I) = \lambda \text{sgn}(x_\lambda(I)) \\ |c(I^c)| \leq \lambda \end{cases} \quad (4.2)$$

O algoritmo de Homotopia procura os vértices da curva  $\lambda \mapsto x_\lambda$ . Começando com  $x_0 = 0$ , em uma iteração  $l$ ,  $\lambda_l = \|c(I)\|_\infty$ , com  $I$  denotando o suporte de  $x_l$ . Depois, o algoritmo calcula uma direção  $d_l$ , onde

$$\begin{cases} A_I^T A_I d_l(I) = \text{sgn}(c_l(I)) \\ d_l(I^c) = 0 \end{cases} \quad (4.3)$$

$A_I$  denota a matriz  $[A_{i_1}, \dots, A_{i_r}]$ , onde cada  $i_r \in I$  e cada  $A_{i_r}$  é uma  $i_r$ -ésima coluna de  $A$ . Da mesma maneira,  $d_l(I)$  é o vetor calculado após remover todos os elementos de  $d_l(i)$  tais que  $i \notin I$ .

A magnitude  $\gamma_l$  do passo  $d_l$  é calculada como o menor valor em que a equação (4.2) não seja mais válida, ou seja, para  $x_{l+1} = x_l + \gamma_l d_l$ , teremos que pelo menos uma das seguintes condições será satisfeita:

(i) para algum  $i \in I$ ,  $|(c_l)_i| > \lambda$ . Isso ocorre para  $\gamma_l = \gamma_l^+$ , onde

$$\gamma_l^+ = \min_{i \in I^c} \left\{ \frac{\lambda_l - c_l(i)}{1 - a_i^T v_l}, \frac{\lambda_l + c_l(i)}{1 + a_i^T v_l} \right\}$$

Onde  $v_l = A_I d_l(I)$ .

(ii) para algum  $i \in I$ ,  $(x_{l+1})_i = 0$ , o que equivale a  $\gamma = \gamma^-$ , onde

$$\gamma_l^- = \min_{i \in I} \left\{ \frac{-x_l(i)}{d_l(i)} \right\}$$

Então calculamos  $\gamma = \min\{\gamma_l^-, \gamma_l^+\}$ . O algoritmo termina quando  $c_l = 0$  ou quando  $\lambda_l = \|c_l\|_\infty \leq \lambda_{l-1} = \|c_{l-1}\|_\infty$

O Algoritmo 22 mostra o funcionamento passo a passo.

---

**Algoritmo 1:** Homotopia

---

**Entrada:**  $A$  matriz  $m \times n$ ,  $y \in \mathbb{R}^m$

**Saída:**  $x \in \mathbb{R}^n$  esparso com  $Ax = y$

```
1 início
2    $\lambda_0 = -1$ 
3    $l = 1$ 
4    $x_l = 0, c_l = -A^T y, \lambda_l = \|c_l\|_\infty$ 
5    $I = \text{supp}(c_l)$ 
6   enquanto  $\lambda_l \geq 0$  e  $\lambda_l > \lambda_{l-1}$  faça
7        $d_l = (A_I^T A_I)^\dagger \text{sgn}(c_l(I)) = A_I^\dagger (A_I^\dagger)^T \text{sgn}(c_l(I))$ 
8        $v_l = A_I d_l(I)$ 
9       Calcular  $\gamma_l^+$  como o valor mínimo da família  $\{\frac{\lambda_l - c_l(i)}{1 - a_i^T v_l}\}_{i \in I^c}$  e  $i^+$  o índice onde
          esse valor é atingido.
10      Calcular  $\gamma_l^-$  como o valor mínimo da família  $\{\frac{-x_l(i)}{d_l(i)}\}_{i \in I}$  e  $i^-$  o índice onde esse
          valor é atingido.
11       $\lambda_l = \min\{\lambda_l^-, \lambda_l^+\}$ 
12       $i = \min\{i^-, i^+\}$ 
13      se  $i \in I$  então
14          | Adicionar  $i$  a  $I$ 
15      senão
16          | Remover  $i$  de  $I$ 
17      fim
18       $x_{l+1} = x_l + \lambda_l d_l$ 
19       $l = l + 1$ 
20       $c_l = A^T(Ax_l - y)$ 
21 fim
22 fim
```

---

Como o programa calcula a pseudoinversa de  $A_I$  em cada iteração e entre uma iteração e

a seguinte  $A_I$  tem uma coluna a mais ou a menos que a  $A_I$  anterior, fizemos uma adaptação do algoritmo recursivo de pseudo-inversa [8] para calcular  $A_I^\dagger$ , a pseudo-inversa de  $A_I$ , de uma forma mais eficiente quando adicionamos um elemento a  $I$ , usando a pseudo-inversa anterior.

Quando removemos um elemento de  $I$ , calculamos  $A_I^\dagger$  da maneira usual.

## 4.1 Exemplo

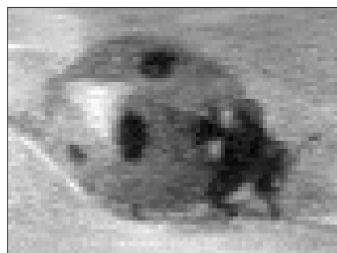
Usamos o algoritmo de homotopia para recuperar uma imagem usando diferentes taxas de compressão.

Fixada uma taxa de compressão  $r$ , calculamos  $m = (1 - r) \cdot n$ . Dada uma imagem, calculamos um frame com a transformada do cosseno da imagem e depois obtemos um vetor  $x$  empilhando todas as linhas do frame. Em seguida, calculamos  $y = Ax$ , onde os elementos de  $A$  são independentes e gerados pela distribuição normal de média 0 e variância 1. Depois encontramos uma aproximação  $\tilde{x}$  para  $x$  usando o algoritmo de homotopia.

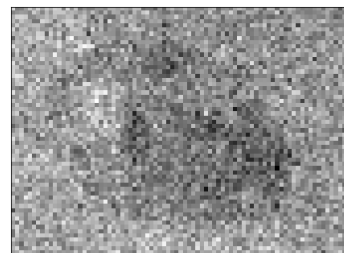
A Figura 4.1 mostra uma imagem recuperada pelo algoritmo de homotopia usando uma taxa de compressão de 20%. Note que o esse algoritmo apresenta melhor resultado que o método de mínimos quadrados com a mesma taxa de compressão.



(a)



(b)



(c)

Figura 4.1: **(a)** Imagem original; **(b)** Imagem recuperada com o algoritmo de homotopia usando taxa de compressão de  $r = 20\%$ ; **(c)** Imagem recuperada usando o método de mínimos quadrados a  $r = 20\%$ . Adaptado de [https://commons.wikimedia.org/wiki/File:5-stip\\_LHB\\_09.jpg](https://commons.wikimedia.org/wiki/File:5-stip_LHB_09.jpg)

# Capítulo 5

## O modelo *cross-and-bouquet*

O modelo *cross-and-bouquet* usa *Compressive Sensing* para classificar imagens, ou seja, dada uma amostra  $y$  e uma família de imagens  $(a_i)_{i=1}^n$ , encontrar qual  $a_i$  é mais próximo de  $y$ .

Podemos interpretar o problema  $(P_0)$  da seguinte forma: dado um vetor  $y \in \mathbb{R}^m$ , quais colunas  $a_i \in \mathbb{R}^m$  de  $A$  melhor representam  $y$ ? Se  $y$  puder ser escrito como  $y = Ax$  com  $x$  esparso, podemos assumir que a coluna  $a_i$  mais próxima de  $y$  é aquela onde  $x_i$  possui maior valor absoluto, ou seja,

$$i = \operatorname{argmax}_{j=1,\dots,n} |x_j|$$

Como vimos, CS garante que conseguimos encontrar  $x$  esparso resolvendo  $(P_1)$  apenas quando  $A$  é incoerente. Em aplicações, como em processamento de imagens, os vetores  $a_i$  não são “muito diferentes” entre si, por isso não podemos assumir que  $A$  é incoerente. Então formularemos o problema de uma forma um pouco diferente:

Dado  $y \in \mathbb{R}^m$  e  $A$  uma matriz  $m \gg n$  (ou seja, o número de amostras é pequeno se comparado à dimensão de cada amostra),

$$\min_{x \in \mathbb{R}^n} \|x\|_1 + \|e\|_1 \text{ sujeito a } y = Ax + e \quad (\tilde{P}_1)$$



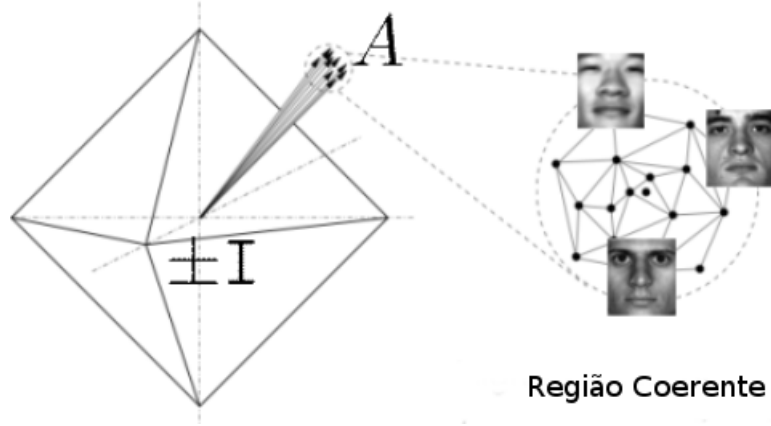


Figura 5.1: Modelo *cross-and-bouquet*. Imagem representando as colunas de  $[A|I]$  onde cada coluna de  $A$  pode representar uma imagem, por exemplo. Adaptado de [18].

o que é equivalente a encontrar um vetor  $c = \begin{bmatrix} x \\ e \end{bmatrix}$ , com  $x \in \mathbb{R}^n$  e  $e \in \mathbb{R}^m$ , onde  $c$  resolve o problema

$$\min_{c \in \mathbb{R}^{n+m}} \|c\|_1 \text{ sujeito a } y = \begin{bmatrix} A & I \end{bmatrix} c \quad (\tilde{P}_1)$$

onde  $I$  é a matriz identidade  $m \times m$ .

Como  $I$  é incoerente, pois é ortonormal, e possui muito mais colunas que  $A$ , é razoável supor que  $\begin{bmatrix} A & I \end{bmatrix}$  também seja incoerente e, neste caso, a solução  $c$  de  $(\tilde{P}_1)$  seria esparsa.

Identificamos a amostra  $y$  com o vetor  $a_i$  onde

$$i = \operatorname{argmax}_{i=1, \dots, n} |c_i|$$

Essa é a ideia do modelo *cross-and-bouquet* [17]. O modelo tem esse nome porque as colunas de  $I$  são ortonormais, lembrando uma cruz e as colunas de  $A$  estão próximas, lembrando um buquê, como mostra a Figura 5.1.

## 5.1 Desempenho

Dada uma imagem  $y$ , poderíamos ter identificado a imagem com a amostra  $a_i$  que maximiza a correlação entre  $a_i$  e  $y$ . Desenvolvemos um programa para classificar imagens do olho usando correlação e *cross-and-bouquet*.

Coletamos imagens de uma pessoa olhando para pontos dispostos numa grade  $7 \times 7$  no monitor, como descrito no Capítulo 6. Calculamos a matriz  $A$  usando a primeira imagem registrada para cada amostra e selecionamos aleatoriamente 5 imagens dos olhos correspondendo a cada posição da grade. Para cada imagem  $y$ , estimamos a posição da grade correspondente a  $y$  como:

- **Correlação:** usando correlação, estimamos a posição cuja coluna  $a_i$  apresenta maior correlação em módulo entre  $a_i$  e  $y$ ;
- ***Cross-and-bouquet*:** identificamos a coluna  $a_i$  mais próxima de  $y$  usando o modelo *cross-and-bouquet*.

Testamos para diferentes tamanhos de imagem e, calculamos a taxa de acertos (ou seja, a quantidade de vezes em que o algoritmo identificou a posição correta do olhar dividido pelo número de imagens usadas,  $5 \times 7 \times 7 = 245$ ). A Tabela 5.1 mostra os resultados para diferentes tamanhos de imagem. O melhor resultado calculado usando correlação é bem inferior aos resultados calculados usando *Compressive Sensing*.

proporções da imagem	correlação	<i>cross-and-bouquet</i>
$640 \times 480$	14, 29%	—
$320 \times 240$	12, 25%	—
$160 \times 120$	10, 20%	—
$80 \times 60$	8, 16%	90, 61%
$40 \times 30$	6, 12%	91, 02%
$20 \times 15$	2, 04%	86, 94%

Tabela 5.1: Desempenho dos algoritmos de correlação e *cross-and-bouquet* para identificar imagens. Note que o desempenho é muito inferior quando usamos correlação.

## Capítulo 6

# Desenvolvimento de um rastreador de olhar usando CS

Alguns algoritmos de rastreamento de olhar usam técnicas de processamento de imagens para identificar a posição da pupila. Essa tarefa é dificultada pela oclusão parcial da pupila pelos cílios. Em vez de estimar a posição da pupila, podemos comparar a imagem com amostras de imagens do olho onde a posição do olhar é conhecida e depois estimar o olhar a partir das posições de olhar das amostras mais parecidas com a imagem atual.

Usamos o modelo *cross-and-bouquet* para comparar imagens das amostras e estimar a posição do olhar.

Desenvolvemos um programa para estimar o olhar do usuário, ou seja, estimar qual coordenada  $(x, y)$  na tela o usuário está olhando. Para isso, usamos câmera da Pupil Labs para coletar imagens do olho direito. Essa câmera fica acoplada à cabeça e registra imagens em infravermelho <sup>1</sup>. O programa é dividido em duas etapas principais:

- **Calibração:** exibimos alvos em posições específicas na tela e coletamos uma imagem do olho por alvo, assumindo que o usuário está olhando para o alvo.
- **Rastreamento:** para cada frame  $f$ , encontramos as amostras mais parecidas com  $f$ .

---

<sup>1</sup><https://github.com/pupil-labs/pupil/wiki/Environment>



Figura 6.1: O programa exibe uma imagem do olho antes da coleta, para o usuário ajustar a câmera de forma que o olho inteiro seja registrado.

Estimamos a posição do olhar como a média das posições da na tela correspondentes às amostras selecionadas.

Essas etapas são descritas mais detalhadamente abaixo.

## 6.1 Coleta de amostras

Fizemos o experimento com seis pessoas para avaliar o desempenho do rastreador de olhar. Durante o experimento o participante ficou sentado a  $56,5\text{cm}$  de distância do monitor, com o olho direito alinhado com o centro da tela. Para evitar movimentos da cabeça durante a coleta das imagens, a pessoa ficou com o queixo apoiado sobre o punho e cotovelo correspondente apoiado na mesa. Registramos imagens do olho direito usando uma câmera do Pupil Labs, acoplada à cabeça, que grava imagens em infravermelho.

A câmera do Pupil Labs estava ligada a um suporte de formato parecido com uma armação de óculos. Por esse motivo, quem usa óculos teve que retirar os óculos para participar do experimento.

Antes de iniciar a coleta, mostramos a imagem da câmera do olho para a pessoa ajustar a câmera de forma que o olho inteiro esteja visível na imagem, como mostra a Figura 6.1.

Foram exibidos aleatoriamente 49 pontos dispostos numa grade  $7 \times 7$  de pontos igualmente

espaçados, onde a distância entre cada lado da grade e o canto correspondente da tela corresponde a um grau do campo visual.

Mostramos cada alvo individualmente durante dois segundos e o usuário deveria olhar para o ponto, como mostra a Figura 6.2. Mostramos os pontos aleatoriamente para evitar aprendizado do usuário, ou seja, evitar que o usuário olhe para o ponto correspondendo à amostra seguinte antes de terminar a coleta da amostra atual. Apesar de mostrar cada ponto durante dois segundos, coletamos imagens apenas após o primeiro segundo porque durante os primeiros instantes ocorre a sacada, ou seja, o movimento do olhar até a região de interesse, e queremos registrar apenas imagens correspondendo à fixação, ou seja, quando a pessoa está realmente olhando para o ponto.

Usamos um monitor de  $37,3cm$  de largura e altura de  $32cm$ , com resolução  $1280 \times 1024$ . A coleta durou menos de 10 minutos por participante.

## 6.2 Rastreamento de olhar

Após terminar a coleta das imagens, estimamos a posição do olhar para imagens correspondendo a todos os pontos da grade  $7 \times 7$ . Como todas imagens registradas correspondem aos pontos da grade, já sabemos a posição do olhar em cada um deles, então usaremos a posição exata e a posição estimada para calcular o erro como a distância euclidiana entre elas. Usaremos a posição de cada alvo em graus horizontal e vertical, e não em *pixels* na tela.

Para cada posição na grade, consideramos a primeira imagem coletada como a amostra correspondente àquela posição na tela, e estimaremos a posição do olhar para cinco das outras imagens, selecionadas aleatoriamente, comparando esta com todas as amostras, como mostra a Figura 6.3.

Pelo modelo *cross-and-bouquet*, para cada vetor correspondendo à imagem  $y$ , calculamos um vetor  $x = (c, e)$  tal que  $y = Ac + e$ . Podemos assumir que as amostras mais parecidas com  $y$ , são as colunas  $a_i$  de  $A$  onde  $c_i$  possuem maior valor em módulo. Assumindo isso,



Figura 6.2: Durante uma coleta, a pessoa deve olhar para os alvos que são exibidos em posições diferentes em uma grade  $7 \times 7$ .

$$A = \left[ \begin{array}{c|c|c|c|c} \text{eye1} & \text{eye2} & \text{eye3} & \dots & \text{eyeN} \end{array} \right]$$

$$y = \left[ \text{eyeY} \right]$$

Figura 6.3: Dada uma matriz com as amostras e uma imagem  $y$ , identificamos as amostras mais parecidas com  $y$  usando o modelo *cross-and-bouquet*, depois estimamos o olhar como a média ponderada das posições de olhar dessas amostras.

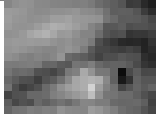
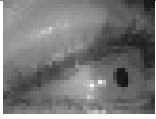
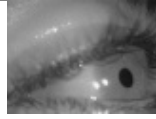
Dimensões das imagens	$15 \times 20$	$30 \times 40$	$60 \times 80$
Erro	$1,471 \pm 2,398$	$1,113 \pm 2,291$	$1,053 \pm 2,279$
Tempo de processamento	$0,035s$	$0,215s$	$5.311s$
Imagem			

Tabela 6.1: Erros na estimação do olhar para diferentes tamanhos de amostras. Cada coluna representa o tamanho das imagens usadas, e cada célula representa o erro médio  $\pm$  o desvio padrão em graus. Vemos também o tempo médio para estimar o olhar de cada imagem. A última linha mostra exemplos de imagens usadas nas respectivas dimensões.

estimamos a posição do olhar para a imagem  $y$  como a média ponderada das posições do olhar das três imagens mais parecidas com  $y$ , onde  $|c_i|$  são os pesos.

Antes de comparar as imagens usando o modelo *cross-and-bouquet* reduzimos as imagens aplicando a pirâmide gaussiana 3, 4 ou 5 vezes, resultando em imagens  $60 \times 80$ ,  $40 \times 30$  e  $20 \times 15$ , respectivamente. Fazemos isso para evitar erros de memória durante a execução do programa.

Após calcular a matriz  $A$ , estimamos o olhar de cada imagem. A entrada do algoritmo será então a imagem e a saída será a posição do olhar em graus do campo de visão do usuário.

## 6.3 Resultados

A Tabela 6.1 mostra a média dos erros (conhecida como acurácia) e o desvio padrão (a precisão), considerando diferentes tamanhos de imagens. Podemos observar que o erro é menor se usamos imagens maiores para estimar o olhar. Para imagens  $60 \times 80$ , obtemos uma acurácia de 1,053 e precisão de 2,279, semelhantes aos resultados obtidos pelo rastreador Tobii EyeX, que tem acurácia de 1,42 e precisão de 1,70, segundo [13]. A Tabela 6.1 indica também o tempo médio para estimar o olhar em cada imagem.



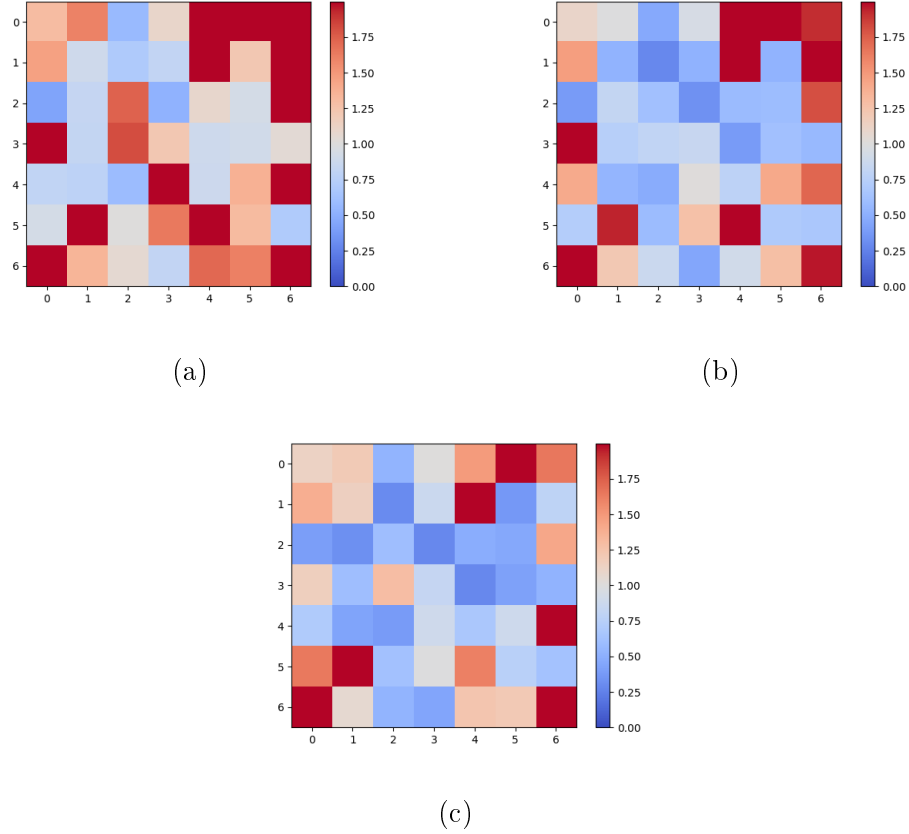


Figura 6.4: Erros médios para cada posição na grade, usando amostras correspondentes a uma grade  $7 \times 7$ . Em **(a)** As imagens usadas têm dimensão  $20 \times 15$ , em **(b)** as imagens são  $40 \times 30$  e em **(c)** as imagens são  $60 \times 80$ .

A Figura 6.4 mostra a distribuição dos erros nos pontos da grade para diferentes tamanhos de imagem, após reduzir as imagens originais aplicando a pirâmide gaussiana 3 vezes (resultando em uma imagem  $60 \times 80$ , 4 vezes (resultando em imagens  $30 \times 40$  e 5 vezes (resultando em imagens  $15 \times 20$ ). Podemos notar na última imagem que os erros são maiores nos cantos da grade.

A Figura 6.5 mostra a acurácia para cada participante. Observando os dados e as imagens registradas durante o experimento, notamos que se o participante piscar durante a coleta das amostras, o resultado será pior. O participante 2, por exemplo, piscou algumas vezes

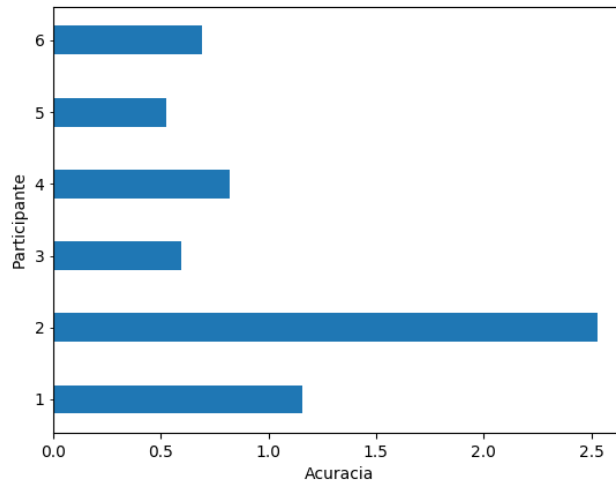


Figura 6.5: A figura mostra a acurácia para diferentes participantes usando o algoritmo para estimar olhar com imagens  $60 \times 80$ . A acurácia para o participante 2 foi pior porque ele piscou durante a coleta das amostras.

durante a coleta de algumas amostras (que eram usadas para estimar o olhar nas outras imagens), acreditamos que por isso o resultado foi pior que os dos outros participantes.

Uma limitação do experimento foi que todas as imagens correspondentes a um ponto na grade foram coletadas durante um intervalo de um segundo, então elas talvez sejam mais parecidas entre si do que seriam se fossem registradas em momentos diferentes.

Apesar das limitações do experimento, o rastreador apresentou um desempenho semelhante ao de um rastreador comercial.

# Capítulo 7

## Conclusão

Neste trabalho estudamos conceitos de rastreamento de olhar, a teoria de *Compressive Sensing* (CS) e desenvolvemos um programa de rastreamento de olhar.

CS é útil para recuperar sinais esparsos, resolvendo com alta probabilidade um problema NP. Uma variação do CS pode ser usada para comparar imagens, usamos esta variação para construir um rastreador de olhar.

Elaboramos um experimento para testar a precisão e acurácia do rastreador e notamos que o programa apresenta algumas limitações, por exemplo, o programa não consegue estimar o olhar para determinada região se o participante piscar durante a coleta da amostra correspondente. Apesar das limitações, o experimento apresentou um resultado semelhante ao de um rastreador comercial.

Este trabalho contribuiu para a formação do aluno em Matemática Aplicada pois, durante o ano o aluno estudou a técnica de *Compressive Sensing*, que apresenta resultados não triviais, e a aplicou em um problema de processamento de imagens.

# Referências Bibliográficas

- [1] Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.
- [2] Mark A Davenport, Marco F Duarte, Yonina C Eldar, and Gitta Kutyniok. Introduction to compressed sensing. *Preprint*, 93(1):2, 2011.
- [3] D Donoho and Y Tsaig. Fast solution of l1-norm minimization problems when the solution may be sparse, 2006. *Preprint*, 1(2).
- [4] Andrew Duchowski. *Eye tracking methodology: Theory and practice*, volume 373. Springer Science & Business Media, 2007.
- [5] Andrew T Duchowski. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4):455–470, 2002.
- [6] Massimo Fornasier. Numerical methods for sparse recovery. *Theoretical foundations and numerical methods for sparse recovery*, 14:93–200, 2010.
- [7] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Springer, 2013.
- [8] TNE Greville. The pseudoinverse of a rectangular matrix and its statistical applications. *Amer. Statist. Assoc., Proc. Soc. Statist. Sect*, pages 116–121, 1958.

- [9] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.
- [10] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. Eyeswipe: Dwell-free text entry using gaze paths. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1952–1956. ACM, 2016.
- [11] Gitta Kutyniok. Theory and applications of compressed sensing. *GAMM-Mitteilungen*, 36(1):79–101, 2013.
- [12] Dongheng Li, David Winfield, and Derrick J Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 79–79. IEEE, 2005.
- [13] R Lima, A Borges, and A Kurauchi. A comparison between intel realsense and tobii eyex for gaze estimation. 2016.
- [14] Robert Gabriel Lupu and Florina Ungureanu. A survey of eye tracking methods and applications. *Bul Inst Polit Iasi*, pages 71–86, 2013.
- [15] Jakob Nielsen. Fancy formatting, fancy words= looks like a promotion= ignored. *UseIt.com Alerbox*, 2007.
- [16] Roberto Valenti, Jacopo Staiano, Nicu Sebe, and Theo Gevers. Webcam-based visual gaze estimation. In *International Conference on Image Analysis and Processing*, pages 662–671. Springer, 2009.
- [17] J Wright and Y Ma. Dense error correction via l1-minimization (2008)(preprint).
- [18] Allen Y Yang, Arvind Genesh, Zihan Zhou, S Shankar Sastry, and Yi Ma. A review of fast l (1)-minimization algorithms for robust face recognition. Technical report, DTIC Document, 2010.

- [19] Y Zhang. Theory of compressive sensing via  $l_1$ -minimization: a non-rip analysis and extensions, rice university, houston. Technical report, TX, Tech. Rep, 2008.