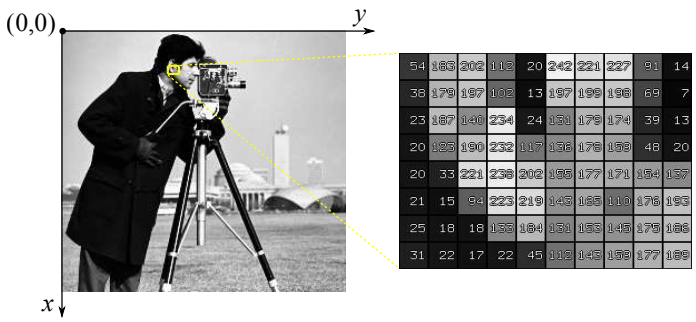
- Uma imagem em níveis de cinza é comumente representada por uma matriz de inteiros sem sinal de 8 bits, ou seja, contendo intensidades de 0 (preto) a 255 (branco):



Para tornar o desenvolvimento mais interessante em Prolog, pode-se representar a imagem como uma lista de pixels, sendo cada pixel definido por uma coordenada e uma intensidade ou (x, y, I). O exemplo a seguir ilustra, por exemplo, o resultado de uma regra chamada `coord` que transforma uma lista de listas M (na forma de uma matriz) em uma lista de pixels S:

```
matrix([[50,10,30],
        [10,20,40]]).

?- matrix(M), coord(M,S).

M = [[50,10,30],[10,20,40]]
S = [(0,0,50),(0,1,10),(0,2,30),(1,0,10),(1,1,20),(1,2,40)]
```

Todas as demais operações são realizadas a partir desta lista de pixels ou simplesmente imagem. Para este trabalho, são utilizadas apenas **imagens binárias**, ou seja, constituídas exclusivamente por *zeros* (pixels pretos) e *uns* (pixels brancos). Considere também que há apenas um objeto (componente conexo) em cada imagem.

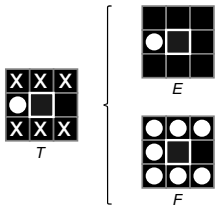
Algumas regras para auxiliar no processamento, análise e reconhecimento de imagens podem ser consultadas e livremente utilizadas a partir da implementação inicial disponibilizada nos *materiais* a seguir.

- Materiais:
  - `img.pl` – funcionalidades básicas para processamento de imagens:
    `https://www.inf.ufsc.br/~alexandre.silva/courses/17s2/ine5416/exercicios/t3/img.pl`
  - `imgs.zip` – conjunto de imagens de teste em PGM (em texto plano que pode ser lido pelo `readPGM` e visualizada pelo `viewPGM` do `img.pl`):
    `https://www.inf.ufsc.br/~alexandre.silva/courses/17s2/ine5416/exercicios/t3/imgs.zip`
    - Observação: imagens do tipo PGM são naturalmente visualizadas no Linux; caso utilize outro sistema, faça upload em:
      `http://faz.retina.ufsc.br/pgmview.html`

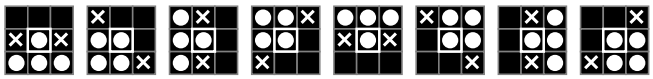
- *Hit-or-miss* e afinamento

  O operador *hit-or-miss* consiste na detecção de um padrão T em uma imagem binária A, sendo T composto por dois elementos estruturantes, E e F, em que $E \subseteq F$.

  Exemplo: padrão para casamento de um pixel de fundo (0) que tenha seis pixels "tanto faz" (0 ou 1) acima e abaixo, um pixel de objeto (1) do lado esquerdo e um pixel de fundo (0) do lado direito.



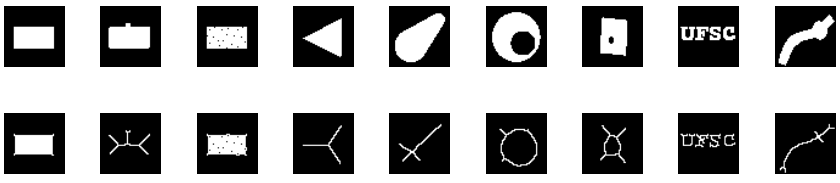
  O *afinamento* consiste em uma técnica iterativa para redução de um objeto até resultar em linhas com um único pixel de espessura. Segue um algoritmo de *afinamento* utilizando *hit-or-miss*, com base em um padrão (círculos exigem 1, vazios exigem 0, e "×" podem ser 0 ou 1) em oito orientações:



AFINAMENTO(A)

1: $E \leftarrow [[0,0,0],[0,1,0],[1,1,1]]$
2: $F \leftarrow [[0,0,0],[1,1,1],[1,1,1]]$
3: $T_0 \leftarrow$ cria padrão *hit-or-miss* usando E e F ▷ sugestão em Prolog: `T0 = [[0,0,0],[x,1,x],[1,1,1]].`
4: **faça**
5: **para cada** $\alpha \in [0^o, 45^o, 90^o, 135^o, 180^o, 225^o, 270^o, 315^o]$ **faça**
6: $T \leftarrow$ rotacão de T_0 em α ▷ em termos práticos, criação prévia e uso dos 8 padrões
7: $P \leftarrow$ pixels de A que casam com o padrão T ▷ usando regras em Prolog (não é necessária morfologia)
8: Atualização de A para trocar seus pixels P para 0 (zero) ▷ passo de afinamento da imagem A
9: **enquanto** $P \neq \emptyset$

- Exemplos de resultados:



- Materiais complementares e aplicações:
  - `https://homepages.inf.ed.ac.uk/rbf/HIPR2/hitmiss.htm`
  - `http://www.inf.u-szeged.hu/~palagyi/skel/skel.html`

- Pede-se a implementação, em Prolog, do:

  - Operador *hit-or-miss* com padrão formado por E e F (considere apenas o tamanho 3×3) aplicado à imagem A, de modo a produzir, como resposta, uma imagem R:
    `hitmiss(A,E,F,R) :-` _____

  - Operador de *afinamento* aplicado à imagem A, de modo a produzir, como resposta, uma imagem R:
    `afinamento(A,R) :-` _____

  - Pesquise e implemente uma outra aplicação de *hit-or-miss* (detecção de pontos finais, de pontos de cruzamentos, de contornos, de fecho convexo, entre outras). Referências:
    * `http://www.cs.tau.ac.il/~turkel/notes/hit-or-miss.pdf`
    * `https://books.google.com.br/books?id=2lbAoonEVrEC&pg=PA182&lpg=PA181&ots=wVlw9Q4ZDG&focus=viewport&dq=thinning+morphology&hl=pt-BR#v=onepage&q=thinning%20morphology&f=false`

    `aplicacao(A,R) :-` _____

  - **(Opcional – pontuação extra)** Faça uma vetorização da imagem resultante do *afinamento*, transformando a matriz para o formato SVG (criação do arquivo `desenhos.svg`), usando a linguagem de desenho do trabalho $t2$ como ferramenta auxiliar.
    `vetorizacao(A) :-` _____

- **Entrega do T_3:**
  - **Prazo:** dia 27nov (segunda) até 23h55
  - **Forma:** Individual ou em grupo de até três alunos
  - **Entrega pelo Moodle:**
    1. Códigos fontes (Prolog)
    2. PDF com explicações e exemplos de aplicação de cada regra e as respostas obtidas