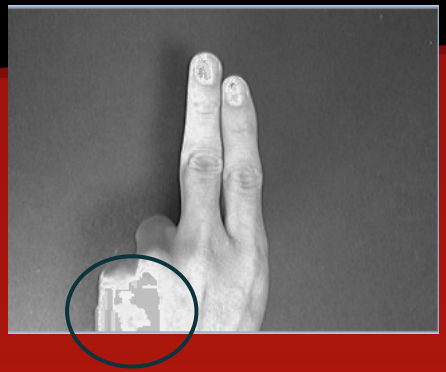
# Projeto de IA

## Aprendizado de Máquina

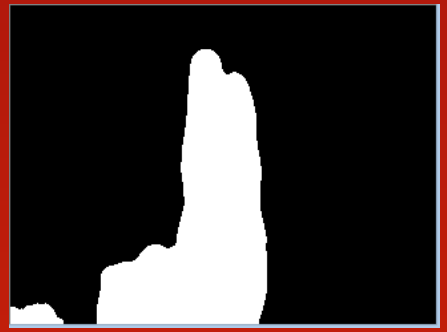
Carlos Mattoso - 1210553
Leonardo Kaplan - 1212509

# ILUMINAÇÃO :x



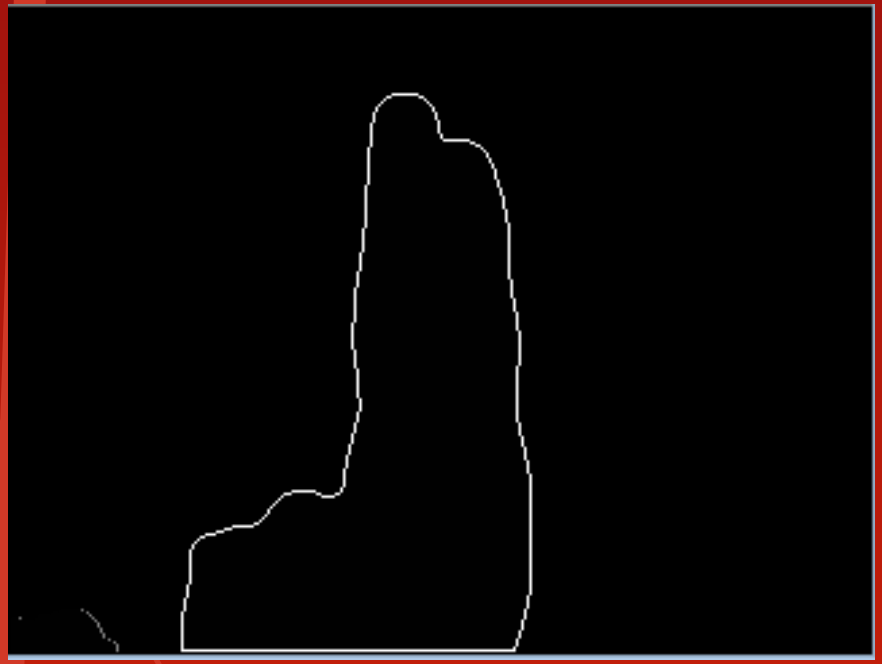
Redução de Luminescência

medianBlur(11)

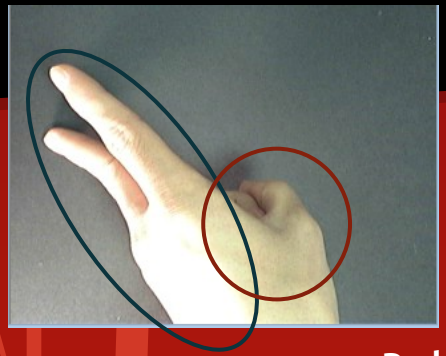


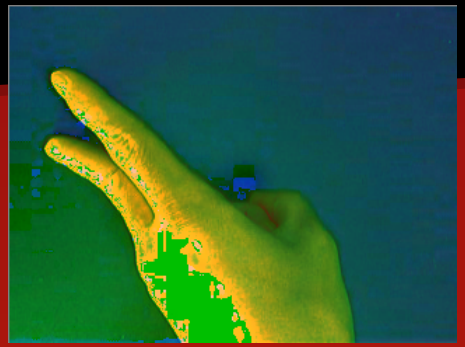
threshold, dilation e erosion

contours



OK! :)

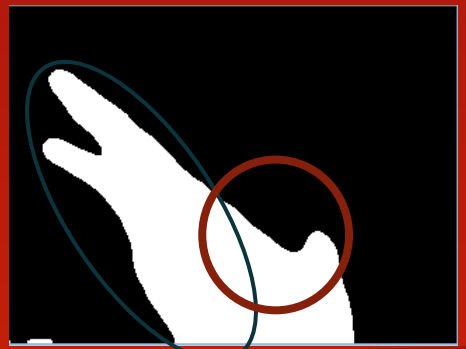




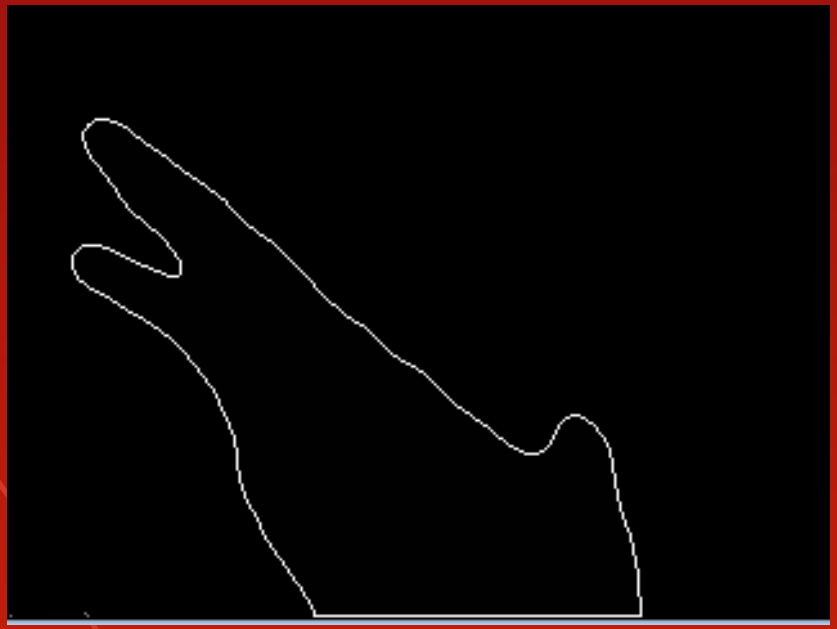
Redução de Luminescência

medianBlur(11)

threshold, dilation e erosion



contours



:(

# Objetivo do trabalho

Treinar uma inteligência artificial para identificar mãos e gestuário manual.

# Técnicas empregadas

OpenCV

Python

Weka

R

# Características selecionadas - 1

Proporção da area do contorno

Proporção da area do retangulo

Aspect Ratio

Ângulo elipse

Profundidade fixpt ( *convexityDefect* )

Coordenadas do centróide

Momentos de Hu 1..7

# Adicionais

Perímetro do Convex Hull

Centro da Elipse

Maior eixo da elipse

Menor eixo da elipse

-> poucos resultados efetivos -> principal: mais lentos

# Resultados

KNN (rápido) ~ 70%

SVM (médio) , NN (muito lento) , Decision Tree (médio) - ~50% :(

- ajustes em função ( svm com polinomial )
- nn com maior taxa de aprendizado
- knn melhora com manhattan

( logs em anexo )

```
=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         628               69.7778 %
Incorrectly Classified Instances       272               30.2222 %
Kappa statistic                          0.66
Mean absolute error                      0.0686
Root mean squared error                  0.2577
Relative absolute error                 34.7253 %
Root relative squared error             82.0151 %
Total Number of Instances              900

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.79      0.028      0.782     0.79      0.786      0.881    0
                 0.72      0.034      0.727     0.72      0.724      0.843    1
                 0.72      0.053      0.632     0.72      0.673      0.834    2
                 0.73      0.029      0.76      0.73      0.745      0.851    3
                 0.68      0.034      0.716     0.68      0.697      0.823    4
                 0.68      0.068      0.557     0.68      0.613      0.806    5
                 0.79      0.034      0.745     0.79      0.767      0.878    6
                 0.67      0.02       0.807     0.67      0.732      0.825    7
                 0.5       0.043      0.595     0.5       0.543      0.729    8
Weighted Avg.    0.698     0.038      0.702     0.698     0.698      0.83

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 79  1  6  2  1  2  6  1  2 |  a = 0
  1 72 10  0  8  3  0  2  4 |  b = 1
  5  7 72  0  0  6  2  4  4 |  c = 2
  5  0  4 73  1 11  4  0  2 |  d = 3
  0  6  4  4 68 12  0  5  1 |  e = 4
  3  4  4  7  6 68  2  1  5 |  f = 5
  4  0  1  5  0  2 79  0  9 |  g = 6
  0  6  4  1 10  3  2 67  7 |  h = 7
  4  3  9  4  1 15 11  3 50 |  i = 8
```

knn - k=1

```
Scheme:weka.classifiers.lazy.IBk -K 4 -W 0 -E -A "weka.core.neighboursearch.LinearN
Relation:     handGestures
Instances:    900
Attributes:   491
[list of attributes omitted]
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 4 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         603               67      %
Incorrectly Classified Instances       297               33      %
Kappa statistic                          0.6288
Mean absolute error                      0.0941
Root mean squared error                  0.2286
Relative absolute error                 47.6454 %
Root relative squared error             72.7484 %
Total Number of Instances              900

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.81      0.05       0.669     0.81      0.733      0.937    0
                 0.73      0.044      0.676     0.73      0.702      0.922    1
                 0.67      0.048      0.638     0.67      0.654      0.913    2
                 0.75      0.043      0.688     0.75      0.718      0.904    3
                 0.68      0.049      0.636     0.68      0.657      0.913    4
                 0.58      0.059      0.552     0.58      0.566      0.86     5
                 0.75      0.03       0.758     0.75      0.754      0.934    6
                 0.67      0.023      0.788     0.67      0.724      0.913    7
                 0.39      0.028      0.639     0.39      0.484      0.798    8
Weighted Avg.    0.67      0.041      0.672     0.67      0.666      0.899

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 81  3  4  5  0  2  5  0  0 |  a = 0
  6 73  7  0  8  4  0  1  1 |  b = 1
  9 10 67  1  0  5  3  3  2 |  c = 2
  4  2  3 75  2  9  4  0  1 |  d = 3
  2  9  2  3 68 11  0  3  2 |  e = 4
  5  2  5 10 16 58  1  1  2 |  f = 5
  5  0  1  8  0  4 75  0  7 |  g = 6
  2  5  6  0 11  1  1 67  7 |  h = 7
  7  4 10  7  2 11 10 10 39 |  i = 8
```

knn - k=4

```
=== Run information ===

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSear
Relation:     handGestures
Instances:    900
Attributes:   666
[list of attributes omitted]
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         623               69.2222 %
Incorrectly Classified Instances       277               30.7778 %
Kappa statistic                          0.6538
Mean absolute error                      0.0698
Root mean squared error                  0.2601
Relative absolute error                 35.3434 %
Root relative squared error             82.7654 %
Total Number of Instances              900

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.79      0.014      0.878      0.79      0.832      0.888    0
                 0.75      0.031      0.75       0.75      0.75       0.859    1
                 0.69      0.034      0.719      0.69      0.704      0.828    2
                 0.75      0.035      0.728      0.75      0.739      0.858    3
                 0.61      0.031      0.709      0.61      0.656      0.789    4
                 0.64      0.068      0.542      0.64      0.587      0.786    5
                 0.68      0.038      0.694      0.68      0.687      0.821    6
                 0.71      0.028      0.763      0.71      0.736      0.841    7
                 0.61      0.069      0.526      0.61      0.565      0.771    8
Weighted Avg.    0.692     0.038      0.701      0.692     0.695      0.827

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 79  1  2  3  0  7  6  0  2 |  a = 0
  1 75  5  0  9  3  1  2  4 |  b = 1
  2  4 69  0  3  7  3  3  9 |  c = 2
  2  0  1 75  2 12  5  1  2 |  d = 3
  0 13  3  2 61 11  1  8  1 |  e = 4
  2  2  2  9  3 64  4  2 12 |  f = 5
  3  0  2 11  0  2 68  0 14 |  g = 6
  0  1  5  1  6  4  1 71 11 |  h = 7
  1  4  7  2  2  8  9  6 61 |  i = 8
```

knn (extras) - k=1

```
=== Run information ===

Scheme:weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -
Relation:     handGestures
Instances:    900
Attributes:   666
[list of attributes omitted]
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

IB1 instance-based classifier
using 1 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         653               72.5556 %
Incorrectly Classified Instances       247               27.4444 %
Kappa statistic                          0.6912
Mean absolute error                      0.0625
Root mean squared error                  0.2456
Relative absolute error                 31.6346 %
Root relative squared error             78.1559 %
Total Number of Instances              900

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.86      0.014      0.887      0.86      0.873      0.923    0
                 0.8       0.029      0.777      0.8       0.788      0.886    1
                 0.72      0.029      0.758      0.72      0.738      0.846    2
                 0.75      0.036      0.721      0.75      0.735      0.857    3
                 0.63      0.024      0.768      0.63      0.692      0.803    4
                 0.72      0.068      0.571      0.72      0.637      0.826    5
                 0.78      0.025      0.796      0.78      0.788      0.878    6
                 0.68      0.029      0.747      0.68      0.712      0.826    7
                 0.59      0.056      0.567      0.59      0.578      0.767    8
Weighted Avg.    0.726     0.034      0.733      0.726     0.727      0.846

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 86  1  1  3  0  5  3  0  1 |  a = 0
  0 80  1  0  7  3  0  4  5 |  b = 1
  3  4 72  0  2  5  3  3  8 |  c = 2
  4  0  2 75  1 13  3  0  2 |  d = 3
  0 10  4  3 63 13  0  5  2 |  e = 4
  2  2  2 12  1 72  2  2  5 |  f = 5
  2  0  1  7  0  2 78  0 10 |  g = 6
  0  2  6  1  7  3  1 68 12 |  h = 7
  0  4  6  3  1 10  8  9 59 |  i = 8
```

knn (extras) - k=1
com manhattan

```
Time taken to build model: 783.55 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         469               52.1111 %
Incorrectly Classified Instances       431               47.8889 %
Kappa statistic                          0.4613
Mean absolute error                      0.1082
Root mean squared error                  0.311
Relative absolute error                 54.7998 %
Root relative squared error             98.9666 %
Total Number of Instances              900

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.69      0.063      0.58      0.69      0.63       0.897    0
                 0.67      0.064      0.568     0.67      0.615      0.927    1
                 0.59      0.059      0.557     0.59      0.573      0.861    2
                 0.36      0.048      0.486     0.36      0.414      0.793    3
                 0.51      0.046      0.58      0.51      0.543      0.851    4
                 0.16      0.03       0.4       0.16      0.229      0.657    5
                 0.67      0.074      0.532     0.67      0.593      0.905    6
                 0.61      0.076      0.5       0.61      0.55       0.856    7
                 0.43      0.08       0.402     0.43      0.415      0.747    8
Weighted Avg.    0.521     0.06       0.512     0.521     0.507      0.833

=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i   <-- classified as
 69  2  7  2  5  0  8  3  4 |  a = 0
  2 67 10  1  2  5  0  9  4 |  b = 1
  8 12 59  2  3  1  4  5  6 |  c = 2
 14  2  2 36  6  4 18  9  9 |  d = 3
  1 13  1 10 51  5  4  9  6 |  e = 4
 12  5 10 13 10 16  6 10 18 |  f = 5
 10  1  3  4  0  2 67  3 10 |  g = 6
  2 12  3  0  6  3  6 61  7 |  h = 7
  1  4 11  6  5  4 13 13 43 |  i = 8
```

nn

# Algoritmo Implementado

KNN -> resultados equivalentes

Feito em R -> lento

```r
knn <- function(train.set, test.set, train.class, k=1, dist=euclidean_dist)
{
  if (is.nan(k)){ k = 1 }
  if(k > nrow(train.set)){ k = nrow(train.set) }


  test.n_row = nrow(test.set)
  train.n_row = nrow(train.set)

  # Array of class predictions for the test set
  test.pred = rep(0, test.n_row)

  # Use this array to store the results of distance functions
  #   No need to reinitialize it, as it gets reset for each
  #   value of i
  d = rep(0, train.n_row)

  # For each test example
  for(i in 1:test.n_row)
  {
    #print(i)

    test.row = as.numeric( test.set[i,] )

    # For each training example
    for(j in 1:train.n_row)
    {

      train.row = as.numeric( train.set[j,] )

      # Get the distance form test.row vector to train.row vector
      #  and save it for later
      d[j] = dist(test.row , train.row)
    }

    # Pick, out of the k classes of nearest neighbors, that which is most common
    #   In case there's a tie, the first one is chosen. To reduce ocurrence of
    #   ties a larger value for k coupled with more training data would help
    test.pred[i] = mode( train.class[ order(d)[1:k] ] )

  }
```

```
> # Start testing knn
> knn_pred = folds(dataset, n_folds=10)
> # If an improvement happens it is displayed
[1] "Fold  1"
[1] "Correct classification rate"
[1] 67.40741
[1] "Confusion Matrix"

best.pred  0  1  2  3  4  5  6  7  8
        0 26  0  0  3  0  1  3  0  0
        1  1 23  2  0  2  0  0  1  1
        2  0  2 22  0  1  1  1  1  2
        3  2  0  0 23  1  1  5  0  1
        4  0  2  1  2 17  3  0  1  1
        5  4  1  0  7  4 21  0  0  4
        6  2  0  2  1  0  0 14  1  2
        7  0  3  1  0  3  1  0 19  1
        8  0  1  2  0  0  0  3  3 17
[1] "Fold  2"
[1] "Correct classification rate"
[1] 68.14815
[1] "Confusion Matrix"

best.pred  0  1  2  3  4  5  6  7  8
        0 25  1  0  1  0  2  1  0  1
        1  0 25  2  0  2  2  0  0  1
        2  0  0 16  1  2  1  1  3  3
        3  1  0  0 16  0  3  4  0  0
        4  0  1  0  0 20  0  0  3  2
        5  1  1  1  6  2 16  0  0  4
        6  2  0  0  1  0  1 19  1  2
        7  0  1  1  0  1  0  1 22  3
        8  1  2  3  0  2  3  6  2 25
[1] "Fold  3"
[1] "Fold  4"
[1] "Correct classification rate"
[1] 69.62963
[1] "Confusion Matrix"

best.pred  0  1  2  3  4  5  6  7  8
        0 24  0  1  1  0  2  2  0  0
        1  0 32  0  0  4  1  0  0  2
        2  1  0 21  0  0  2  0  1  3
        3  1  1  0 19  0  0  5  0  0
        4  0  3  2  1 20  3  0  1  1
        5  0  0  0  7  4 13  1  0  1
        6  0  0  1  2  0  2 20  1  1
        7  0  2  0  0  2  1  2 16  1
        8  1  1  3  1  2  1  3  4 23
[1] "Fold  5"
[1] "Fold  6"
[1] "Fold  7"
[1] "Fold  8"
[1] "Fold  9"
[1] "Fold  10"
```

knn (extras) - k=1 - Euclidean

```
> knn_pred = folds(dataset, n_folds=1, dist=manhattan_dist)
[1] "Fold  1"
[1] "Correct classification rate"
[1] 73.7037
[1] "Confusion Matrix"

best.pred  0  1  2  3  4  5  6  7  8
        0 24  0  2  1  0  0  1  0  0
        1  0 29  1  0  0  1  0  2  0
        2  2  1 21  1  1  1  0  1  0
        3  0  0  0 21  1  3  2  0  2
        4  0  2  0  0 26  2  0  0  1
        5  4  0  2  6  4 20  1  1  2
        6  0  0  2  3  0  0 21  0  1
        7  0  1  0  0  2  1  0 17  0
        8  0  1  1  0  0  2  3  6 20
```

knn (extras) - k=1
com manhattan

# Possíveis Melhorias

1. Filtro de pele com HSV
2. Uso de SIFT com Bag of Words

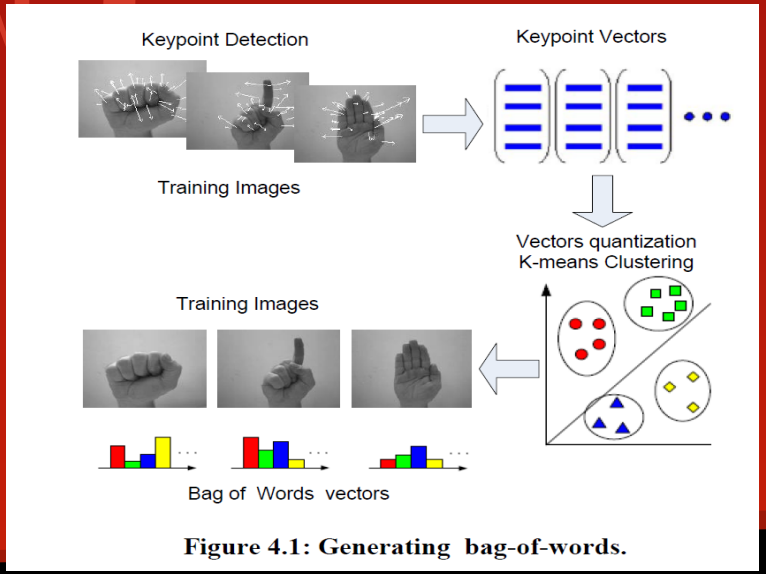



Figure 4.1: Generating bag-of-words.

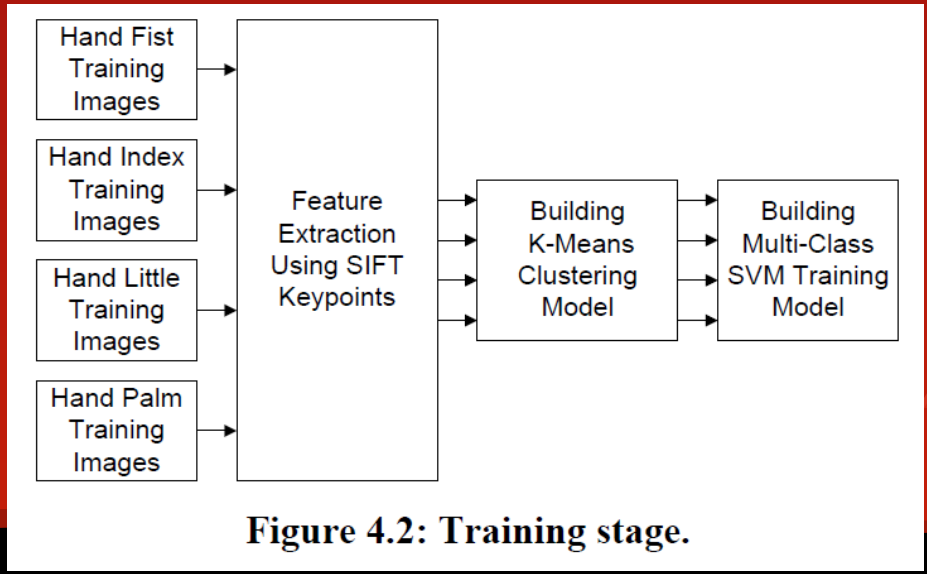



Figure 4.2: Training stage.

# Obrigado!

Dúvidas?