PROFI

Project number:	FP6-511572
Project acronym:	PROFI
Title:	Perceptually-relevant Retrieval Of Figurative Images

Deliverable No: D5.4: Implementation of partial curve matching

Short description:
We implemented the probabilistic algorithm described in deliverable 5.3 for finding partial similarity between two shapes that represent figurative images. Shapes are modeled by sets of polygonal curves. Two variants of partial matching were implemented: partial-partial matching, that is parts of one shape are matched to parts of the other shape, and complete-partial matching, that is the complete first shape is matched against parts of the other shape. In both cases, first some candidate transformations of one shape are found that map the largest possible parts of one shape in the proximity of parts of the other shape. The allowed transformations are translations, rigid motions, homotheties and similarity maps. For each candidate transformation we compute the distance between the one transformed shape and the other shape, and return the transformation yielding the smallest distance together with the distance value. In case of partial-partial matching we also report the relative size of the matched parts and the quality of the match.

Due month:	M18
Delivery month:	M18
Lead partner:	Freie Universität Berlin
Partners contributed:	Freie Universität Berlin
Classification:	RE

Information Society Technologies

Project funded by the European Community under the "Information Society Technologies" Programme

1 Results

1.1 Introduction

In this workpackage we develop and implement algorithms for *matching* parts of two planar shapes that represent figurative images. we assume that shapes are modeled by sets of polygonal curves. As possible class of transformations we consider translations, rigid motions (i.e., translations and rotations), homotheties (i.e., translations and scalings) and similarity maps (i.e., translations, rotations and scalings).

In general, we are given two shapes A and B and a class of allowable transformations and we want to find a transformation, such that the transformed shape B or some parts of it are as close to the shape A as possible. Usually, the quality of match is measured by some distance function, which assigns a number indicating the quality of match to any pair of shapes. In this workpackage we address the problem of *complete-partial matching* (CPM), i.e., matching shape B completely as good as possible to some part of shape A, and the problem of *partial-partial matching* (PPM), i.e., matching some part of B as good as possible to some part of A. Clearly, both partial matching problems CPM and PPM are not uniquely specified since there is a tradeoff between the quality of the match and the size of the matched parts. Which of two criteria is more important, depends on the application. We address this problem by introducing a parameter k which regulates the influence of the quality of match and the matched size on the final distance value. The matched parts, the quality of the match for these parts as well as their relative size with respect to the total size of the shapes are also returned as part of the matching result.

The appendix A describes the programming interface of the partial matching classes, for the complete documentation of the software library please refer to the deliverable report D4.4.

1.2 Matching

For two given shapes we first find some candidate transformations, these transformations are then evaluated according to the appropriate distance function as described in section 1.3. The matching process is implemented according to the description in deliverable report D5.3, section 2.2. We briefly sketch the idea of the matching algorithm:

The transformations are computed based on (local) subsets of vertices. If there exist one or more parts that have a similar counterpart, the transformations computed for these parts will form clusters that yield candidate transformations.

Let S_1 and S_2 be sets of polylines, a vertex $p_{1,0}$ of a polyline $P_1 \in S_1$ and a vertex $p_{2,0}$ of a polyline $P_2 \in S_2$ are randomly chosen in every step. Beginning with these seed vertices an ordered set $\tilde{P}_1 \subseteq P_1$ and an ordered set $\tilde{P}_2 \subseteq P_2$ of vertices or vertex surrogates are generated.

For both polylines starting from the last vertex added to $\tilde{P}$ the distance to the next vertex is computed. If the difference of the distances lies below a certain threshold, both vertices are added, otherwise for the vertex with smaller distance a corresponding vertex surrogate with the same distance is created on the other polyline and this pair is added. For every cardinality $\tilde{m}$ of $\tilde{P}_1$ and $\tilde{P}_2$ the transformation t is computed, such that $\varepsilon = \sum_{i=1}^{\tilde{m}} \|t(p_{1,i}) - p_{2,i}\|^2$ is minimized. The transformation's weight is composed of the length of the covered part of the

polyline and the error ε. For every step the transformation t with the highest weight is handed over to the clustering algorithm.

The largest clusters represent transformations that let the largest parts of the shapes match. In every cluster we keep track of the parts of the shapes contributing to this cluster.

1.3 The Distance Function

We define a resemblance function for two polyline sets S_1, S_2 as follows: The resemblance function ϕ_s for a line segment $s \in S_1$ is defined as

$\phi_s(\lambda) = \max_{g \in S_2} (\alpha_{s,g}(\lambda) \cdot \beta_{s,g})$

with α being the distance factor and β being the slope factor as defined for the complete-complete matching case in deliverable report D4.3, section 2.3.
The weighting function ω (to prevent parts with many parallel lines from dominating over parts with solitary lines) is defined analogously to the resemblance function:

$$\omega_s(\lambda) = \frac{1}{\sum_{g \in S_1} (\alpha_{s,g}(\lambda) \cdot \beta_{s,g})}$$

The directed resemblance measure $\Phi_{\rightarrow}(S_1, S_2)$ is defined by

$$\Phi_{\rightarrow}(S_1, S_2) = \frac{\sum_{s \in S_1} \left(\int_{\lambda=0}^{1} \phi_s(\lambda) \cdot \omega_s(\lambda)\, d\lambda \cdot l_s \right)}{\Omega(S_1)}$$

with l_s being the length of s and $\Omega(S_1)$ being the total weight of S_1:
$\Omega(S_1) = \sum_{s \in S_1} \left(\int_{\lambda=0}^{1} \omega_s(\lambda)\, d\lambda \cdot l_s \right)$.

For the *complete-complete matching* we took a weighted combination of two one-sided resemblance values. Note, that the above definition of the directed resemblance function applied to the complete shapes S_1 and S_2 gives us a matching score of how good the complete shape S_1 is matched to shape S_2 and, therefore, a matching score for the *complete-partial matching*. Since the resemblance function takes values between zero and one, with value of one meaning a perfect resemblance and zero – no resemblance, we have to convert it to a distance function, which would take small values for similar and large value for dissimilar shapes. In the current implementation we define distance as $1 - \Phi_{\rightarrow}(S_1, S_2)$.

For *partial-partial matching*, as mentioned above, for each cluster we keep a record of which parts of the shapes contribute to the transformation defined by this cluster. Let t be a transformation corresponding to a cluster and $S_1^t \subset S_1$ and $S_2^t \subset S_2$ are the parts of the shapes matched by t. Then, we compute the resemblances for the matched parts: $s_1 = \Phi_{\rightarrow}(S_1^t, S_2^t)$ and $s_2 = \Phi_{\rightarrow}(S_2^t, S_1^t)$. However, if we just take the resemblances of parts as the quality of match and compute them for all clusters we would in general get those matching perfectly very small parts of the shapes as the best ones. The first step to avoid this problem and to reduce the number of distance computations, is to consider only the largest clusters.

Among the largest clusters we still want the size of matched parts to affect the matching score. Therefore, we compute a ratio of the matched parts as $r_i = \frac{|S_i^t|}{|S_i|}$ for $i = 1, 2$, where $|S_i|$ denotes the total length of the polylines of the shape S_i and $\left|S_i^t\right|$ denotes the length of

the parts matched by the transformation t, and define a weight factor as $f_i = 1 - (1 - r_i)^k$ (see Figure 1), where k is a (user-defined) parameter; the default value of k is 3 in the current implementation. The maximum of two weight factors $f^* = \max(f_1, f_2)$ is then used to adjust the resemblance value: $s^* = f^* \frac{s_1+s_2}{2}$. Again, the resemblance value needs to be converted to a distance value. The choice of the factor function was motivated by the following consideration: if large parts of at least one shape are matched, we want to leave the resemblance value almost unchanged, and give larger penalties the smaller the matched parts get. With the parameter k the user can control these penalties, if k is large, the resemblance value stays almost unchanged even for small parts, whereas for small values of k the quality of match decreases with the size of matched parts.

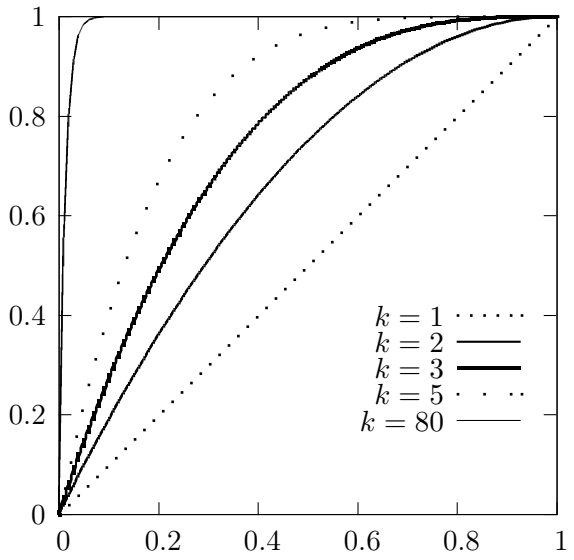

Figure 1: Weight factor parameterized by k.

A more subtle task is to decide whether these parts form an expressive subshape that has been recognized – a match in the PPM / CPM sense – or if they are just independent and do not induce a perceptually relevant resemblance.

2 Deviations from Plan

There have been no deviations from plan.

A Partial matching API

A.1 EuProfi::ProbabilisticMatcherPP Class Reference

`#include <probabilisticMatcher.h>`

A.1.1 Detailed Description

This class implements the probabilistic algorithm for partial matching as described in deliverable D5.3 section 2.2.

The distance function used to evaluate the quality of match is described in deliverable D5.3 section 2.3. The current implementation of the distance function follows the approach of considering only the parts of the shapes that have a correspondence in the other shape for valuating the quality of match and then weighting the result with the relative size of the matched parts.

The full matching result contains the transformation which yields the best match, the distance value corresponding to that transformation, the one-sided resemblances of the matched parts of the two shapes, the relative size of the matched parts and the matched parts of the shapes.

Public Types

- typedef SimilarityInfoP **full_result_type**
- typedef std::vector< full_result_type > **results_type**
- typedef PolylineSet **argument_type**
- typedef PolylineSet2 **argument_type_2**
- typedef char * **argument_type_s**

Public Member Functions

- full_result_type match (argument_type_s shape2, argument_type_s shape1)
- full_result_type match (argument_type shape2, argument_type shape1)
- full_result_type match (argument_type_2 shape2, argument_type_2 shape1)
- full_result_type matchToFixedFirst (argument_type_s shape2)
- full_result_type matchToFixedFirst (argument_type shape2)
- full_result_type matchToFixedFirst (argument_type_2 shape2)
- results_type getResults ()
- double distance (argument_type_s shape2, argument_type_s shape1)
- double distance (argument_type shape2, argument_type shape1)
- double distance (argument_type_2 shape2, argument_type_2 shape1)
- void setFirst (argument_type_s shape1)
- void setFirst (argument_type shape1)
- void setFirst (argument_type_2 shape1)
- double distanceToFixedFirst (argument_type_s shape2)

- double distanceToFixedFirst (argument_type shape2)
- double distanceToFixedFirst (argument_type_2 shape2)
- void clearFirst ()

A.1.2 Member Function Documentation

A.1.2.1 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type_s *shape2*, argument_type_s *shape1*)

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shape data are read from files with names given by shape1 and shape2.

A.1.2.2 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type *shape2*, argument_type *shape1*)

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shapes are given as PolylineSet.

A.1.2.3 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type_2 *shape2*, argument_type_2 *shape1*)

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shapes are given as PolylineSet2.

A.1.2.4 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type_s *shape2*)

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is read from a file with name given by shape2. The first shape should be set previously via call setFirst(shape1).

A.1.2.5 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type *shape2*)

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is given as PolylineSet. The first shape should be set previously via call setFirst(shape1).

A.1.2.6 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type_2 *shape2*)

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is given as PolylineSet2. The first shape should be set previously via call setFirst(shape1).

A.1.2.7 PartialMatchingResults EuProfi::ProbabilisticMatcherPP::getResults ()

Returns all candidate matches found by the last algorithm run, that is, one of the following should be called first: match(), distance(), matchToFixedFirst(), or distanceToFixedFirst().

A.1.2.8 double EuProfi::AbstractMatcher::distance (argument_type_s *shape2*, argument_type_s *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, and computes and returns the distance value corresponding to that transformation.

Shape data are read from files with names given by shape1 and shape2.

A.1.2.9 double EuProfi::AbstractMatcher::distance (argument_type *shape2*, argument_type *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, computes and returns the distance value corresponding to that transformation.

Shapes are given as PolylineSet.

A.1.2.10 double EuProfi::AbstractMatcher::distance (argument_type_2 *shape2*, argument_type_2 *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, computes and returns the distance value corresponding to that transformation.

Shapes are given as PolylineSet2.

A.1.2.11 void EuProfi::AbstractMatcher::setFirst (argument_type_s *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is read from a file with name given by shape1.

A.1.2.12 void EuProfi::AbstractMatcher::setFirst (argument_type *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is given as PolylineSet.

A.1.2.13 void EuProfi::AbstractMatcher::setFirst (argument_type_2 *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is given as PolylineSet2.

A.1.2.14 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type_s *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is read from a file with name given by shape2. The first shape should be set previously via call setFirst(shape1).

A.1.2.15 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is given as PolylineSet. The first shape should be set previously via call setFirst(shape1).

A.1.2.16 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type_2 *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is given as PolylineSet2. The first shape should be set previously via call setFirst(shape1).

A.1.2.17 void EuProfi::AbstractMatcher::clearFirst () [inherited]

Resets the predefined shape.

A.2 EuProfi::ProbabilisticMatcherCP Class Reference

`#include <probabilisticMatcher.h>`

A.2.1 Detailed Description

This class implements the probabilistic algorithm for complete-partial probabilistic matching as described in deliverable D5.3 section 2.2.

The distance function indicates how good the complete first shape is matched to some parts of the second shape.

The full matching result contains the transformation yielding the best complete-partial match, the distance value corresponding to that transformation and the one-sided resemblance values for both shapes.

Public Types

- typedef SimilarityInfoP **full_result_type**
- typedef std::vector< full_result_type > **results_type**
- typedef PolylineSet **argument_type**
- typedef PolylineSet2 **argument_type_2**
- typedef char * **argument_type_s**

Public Member Functions

- full_result_type match (argument_type_s shape2, argument_type_s shape1)
- full_result_type match (argument_type shape2, argument_type shape1)
- full_result_type match (argument_type_2 shape2, argument_type_2 shape1)
- full_result_type matchToFixedFirst (argument_type_s shape2)
- full_result_type matchToFixedFirst (argument_type shape2)
- full_result_type matchToFixedFirst (argument_type_2 shape2)
- results_type getResults ()
- double distance (argument_type_s shape2, argument_type_s shape1)
- double distance (argument_type shape2, argument_type shape1)
- double distance (argument_type_2 shape2, argument_type_2 shape1)
- void setFirst (argument_type_s shape1)
- void setFirst (argument_type shape1)
- void setFirst (argument_type_2 shape1)
- double distanceToFixedFirst (argument_type_s shape2)
- double distanceToFixedFirst (argument_type shape2)
- double distanceToFixedFirst (argument_type_2 shape2)
- void clearFirst ()

A.2.2 Member Function Documentation

A.2.2.1 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type_s *shape2*, argument_type_s *shape1*) [inherited]

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shape data are read from files with names given by shape1 and shape2.

A.2.2.2 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type *shape2*, argument_type *shape1*) [inherited]

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shapes are given as PolylineSet.

A.2.2.3 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::match (argument_type_2 *shape2*, argument_type_2 *shape1*) [inherited]

Finds and returns the best transformation mapping shape2 to shape1, along with the distance value corresponding to that transformation.

Shapes are given as PolylineSet2.

A.2.2.4 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type_s *shape2*) [inherited]

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is read from a file with name given by shape2. The first shape should be set previously via call setFirst(shape1).

A.2.2.5 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type *shape2*) [inherited]

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is given as PolylineSet. The first shape should be set previously via call setFirst(shape1).

A.2.2.6 PartialMatchingResult EuProfi::ProbabilisticMatcherPP::matchToFixedFirst (argument_type_2 *shape2*) [inherited]

Finds and returns the best transformation mapping shape2 to predefined shape, along with the distance value corresponding to the transformation.

Shape is given as PolylineSet2. The first shape should be set previously via call setFirst(shape1).

A.2.2.7 PartialMatchingResults EuProfi::ProbabilisticMatcherPP::getResults () [inherited]

Returns all candidate matches found by the last algorithm run, that is, one of the following should be called first: match(), distance(), matchToFixedFirst(), or distanceToFixedFirst().

A.2.2.8 double EuProfi::AbstractMatcher::distance (argument_type_s *shape2*, argument_type_s *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, and computes and returns the distance value corresponding to that transformation.

Shape data are read from files with names given by shape1 and shape2.

A.2.2.9 double EuProfi::AbstractMatcher::distance (argument_type *shape2*, argument_type *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, computes and returns the distance value corresponding to that transformation.

Shapes are given as PolylineSet.

A.2.2.10 double EuProfi::AbstractMatcher::distance (argument_type_2 *shape2*, argument_type_2 *shape1*) [inherited]

Finds the best transformation mapping shape2 to shape1, computes and returns the distance value corresponding to that transformation.

Shapes are given as PolylineSet2.

A.2.2.11 void EuProfi::AbstractMatcher::setFirst (argument_type_s *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is read from a file with name given by shape1.

A.2.2.12 void EuProfi::AbstractMatcher::setFirst (argument_type *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is given as PolylineSet.

A.2.2.13 void EuProfi::AbstractMatcher::setFirst (argument_type_2 *shape1*) [inherited]

Presets a shape to be compared by consecutive calls of matchToFixedFirst(shape2).

Shape is given as PolylineSet2.

A.2.2.14 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type_s *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is read from a file with name given by shape2. The first shape should be set previously via call setFirst(shape1).

A.2.2.15 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is given as PolylineSet. The first shape should be set previously via call setFirst(shape1).

A.2.2.16 double EuProfi::AbstractMatcher::distanceToFixedFirst (argument_type_2 *shape2*) [inherited]

Finds the best transformation mapping shape2 to the predefined shape, computes and returns the distance value corresponding to the transformation.

The shape is given as PolylineSet2. The first shape should be set previously via call setFirst(shape1).

A.2.2.17 void EuProfi::AbstractMatcher::clearFirst () [inherited]

Resets the predefined shape.

A.3 EuProfi::SimilarityInfoP Struct Reference

`#include <probabilisticMatcher.h>`

Inherits EuProfi::SimilarityInfo.

A.3.1 Detailed Description

Structure SimilarityInfoP contains the result of the partial matching.

It contains the best found transformation and the distance corresponding to that transformation.

In case of partial-partial matching it also contains the information about the relative size of the matched parts of each shape and the one-sided similarity values of the matched parts corresponding to the transformation. Additionally the matched parts of the shapes are returned as sets of polygonal curves. The distance value of the match depends on the distances between the matched parts and on the relative size of the matched parts.

In case of complete-partial matching the ratio-values and the matched parts are unset. The one-sided similarity values are computed for both shapes. The distance value indicates how good the complete first shape is matched to some parts of the second shape.

The transformation can in general be an affine transformation, though typically a similarity transformation is used.

Public Member Functions

- double dist () const

Public Attributes

- double ratioFirst
- double ratioSecond
- double simFirst
- double simSecond
- PolylineSet partsFirst
- PolylineSet partsSecond
- double distance
- AffineTfm atm

A.3.2 Member Function Documentation

A.3.2.1 double EuProfi::SimilarityInfo::dist () const [inline, inherited]

Returns the distance between the matched shapes.

A.3.3 Member Data Documentation

A.3.3.1 double EuProfi::SimilarityInfoP::ratioFirst

Relative size of the matched parts of the first shape.

This value is not set in the result of complete-partial matching.

A.3.3.2 double EuProfi::SimilarityInfoP::ratioSecond

Relative size of the matched parts of the second shape.

This value is not set in the result of complete-partial matching.

A.3.3.3 double EuProfi::SimilarityInfoP::simFirst

One-sided similarity value between the matched parts of the first shape and the matched parts of the second shape.

A.3.3.4 double EuProfi::SimilarityInfoP::simSecond

One-sided similarity value between the matched parts of the second shape and the matched parts of the first shape.

A.3.3.5 PolylineSet EuProfi::SimilarityInfoP::partsFirst

Matched parts of the first shape.

This value is not set in the result of complete-partial matching.

A.3.3.6 PolylineSet EuProfi::SimilarityInfoP::partsSecond

Matched parts of the second shape.

This value is not set in the result of complete-partial matching.

A.3.3.7 double EuProfi::SimilarityInfo::distance [inherited]

Distance between the matched shapes.

A.3.3.8 AffineTfm EuProfi::SimilarityInfo::atm [inherited]

Transformation yielding the best match.