Integrating Multiple Model Views for Object Recognition [*]

Vittorio Ferrari[1], Tinne Tuytelaars[2], Luc Van Gool[1,2]

[1]Computer Vision Group (BIWI), ETH Zuerich, Switzerland

[2]ESAT-PSI, University of Leuven, Belgium

{ferrari,vangool}@vision.ee.ethz.ch, Tinne.Tuytelaars@esat.kuleuven.ac.be

Abstract

We present a new approach to appearance-based object recognition, which captures the relationships between multiple model views and exploits them to improve recognition performance.

The basic building block are local, viewpoint invariant regions. We propose an efficient algorithm for partitioning a set of region matches into groups lying on smooth surfaces (GAMs). During modeling, the model views are connected by a large number of region-tracks, each aggregating image regions of a single physical region across the views. At recognition time, GAMs are constructed matching a test image to each model view. The consistency of configurations of GAMs is measured by exploiting the model connections. The most consistent configuration, covering the object as completely as possible is found by a genetic algorithm. Introducing GAMs as an intermediate grouping level facilitates decision-making and improves discriminative power.

As a complementary application, we introduce a novel GAM-based two-view filter and demonstrate its effectiveness in recovering correct matches in the presence of up to 96% mismatches.

1. Introduction

In the last few years, object recognition (OR) approaches based on local viewpoint invariant features have become increasingly popular. In the common basic scheme [11, 6, 7, 5], small regions are extracted independently from a model and a test image in a viewpoint invariant way, then characterized by invariant descriptors and finally matched. The object is recognized if a sufficient number of matches is found. The power of these methods is twofold. First, local features bring tolerance to clutter and occlusions. Second, the extractors and descriptors are invariant under affine transformations, allowing large viewpoint changes.

When several model views of an object are available, most systems match the test image to each model view independently, and then just select the best one or consider the total number of matches [4]. Only few earlier works capture and exploit the relationships among the model views. In [5], the model views are first clustered, and links are made between corresponding features in adjacent views. By following the links, a feature from the test image votes for the view to which is matched, and for the adjacent ones. The system gains robustness, because the votes are not dispersed among similar model views. In [8], a higher degree of integration is reached by building a 3D model of the object, prior to recognition. In this paper, we present a new approach which effectively integrates the contributions of multiple model views. It reaches a deeper level of integration than [5], without requiring the construction of a 3D model. This has the advantage that the selection of model views is far less constrained. Indeed, not all features need to be visible in at least two or three views (but we exploit this overlap whenever it occurs). Moreover, there is no danger of degenerate cases such as views showing only a single planar part. As an additional advantage, our method is capable of recognizing objects undergoing non-rigid deformations.

The main ingredient of our approach is the novel concept of a *group of aggregated matches* (GAM). A GAM is a set of region matches between two images, which are distributed over a smooth surface of the object/scene. The GAM idea is similar in spirit to the work of Selinger and Nelson [9], who advocate the benefits of an intermediate perceptual grouping level between primitives and views. Unlike in their work, here the primitives being grouped are region matches, rather than contour fragments. Moreover, GAMs are inherently a two-view concept, whereas contour fragments are defined in individual views. A set of matches, including an *arbitrary* amount of mismatches, can be partitioned into GAMs (section 2). The obtained GAMs have two fundamental properties. First, a GAM is nearly always 'pure', i.e.: composed only of correct matches or only of mismatches. Second, the number of matches in a GAM relates to its probability of being correct. If a GAM is composed of many region matches (typically more than 5), it is very probably correct, whereas if it has only a few, it is usually incorrect. When evaluating the correctness and structure of sets of matches, it is convenient to reason at the higher perceptual grouping level that GAMs offer: no

[*]The authors gratefully acknowledge support by EC project VIBES and the Fund for Scientific Research Flanders (Belgium)

longer consider unrelated region matches, but the collection of GAMs instead. GAMs become the atomic unit, and their sizes is a precious piece of information. Moreover, the computational complexity of a problem can be reduced, because there are far less relevant GAMs than region matches.

We now give an overview of the proposed OR system. During modeling, the model views are densely connected by a number of *region-tracks*. Each region-track connects the image regions of a certain physical surface patch across the views (section 4). At recognition time, we match each model view to the test image and partition the resulting sets of matches into GAMs (section 5). The matching is performed by the method of [4], which densely covers the object with matches and allows non-rigid deformations. The image area covered by the dense set of matches also provides a segmentation of the object. The coherence of a *configuration* of GAMs, possibly originating from different model views, is evaluated using the region tracks that span the model views, and assigned a score (subsection 5.3). We maximize the score function over all possible configurations with a Genetic Algorithm (subsection 5.4). The maximal score represents the system's confidence in the presence of the object and strongly increases in presence of compatible GAMs. In this fashion, the system's ability to discriminate between objects improves over the simple approach of counting the total number of matches to all model views. As another advantage, incorrect GAMs are discovered because they do not belong to the best configuration. These improvements are demonstrated in the result section 6.

As a complementary application, GAMs are very useful also in the context of robust wide baseline stereo matching. In section 3, we propose a GAM-based filter which is largely insensitive to the percentage of correct matches. We show that the filter is capable of recovering correct matches in sets containing up to 96% mismatches.

2. GAM

This section describes an incremental grouping algorithm to partition a set of matches between two images into GAMs. The matches can be generated by any affine invariant region matcher such as [11, 6, 7].

2.1 Affine dissimilarity

The grouping process is driven by the similarity between the geometric (affine) transformations that map the regions from one view to another. Consider 3 points on each region: the center p_0 and two more points p_1, p_2 on the boundary. These points have previously been put in correspondence by the matching algorithm. The following function measures to which degree the affine transformation of a region R is also valid for another region Q (figure 1):

$$\begin{aligned} \mathrm{D}(R,Q) \quad = \quad & \tfrac{1}{6}\big(\textstyle\sum_{i=0..2} \|A_{R_1\to R_2} Q_1^i - Q_2^i\| \\ & + \textstyle\sum_{i=0..2} \|A_{R_2\to R_1} Q_2^i - Q_1^i\|\big) \end{aligned} \qquad (1)$$

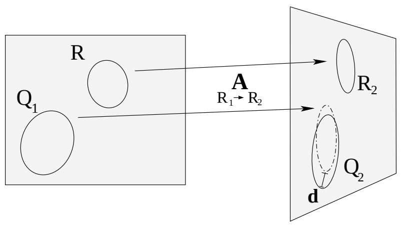

Figure 1: *Affine dissimilarity. d is one term in function (1).*

where $A_{R_a\to R_b}$ is the affine transformation mapping R from view a to view b, and R_v^i is point p_i of region R in view v. By averaging over the two regions, we obtain the *affine dissimilarity* $D_s(R,Q) = \frac{1}{2}(D(R,Q)+D(Q,R))$ between (the affine transformations of) R and Q. This measure is symmetric in the regions *and* in the views. This brings stability and helps dealing fairly with large scale changes. Two region matches have a high dissimilarity if either is a mismatch, or if they lie on different surfaces.

2.2 Constructing GAMs

The matches are partitioned by the following algorithm, which starts a GAM from a single match and then grows it by iteratively adding matches. The algorithm takes as input the set Ω of region matches.

1. A region match is removed from Ω and used to create a new GAM.
2. Search Ω for a region with affine dissimilarity to the GAM below a certain threshold [1]. The search proceeds from the region which is spatially closest to the GAM, to the spatially farthest. The spatial distance of a region to the GAM is the average Euclidean distance to the composing regions, measured in the first view. The affine dissimilarity between a region and the GAM is the weighted mean of the affine dissimilarities to each region in the GAM. The weights are set inversely proportional to the square of the distances between the regions , and sum up to 1.
3. As soon as a suitable region is found, it is added to the GAM and the search stops. The region is removed from Ω, and the algorithm iterates to 2. If no such region is found, the current GAM is closed. The algorithm goes back to 1, where a new GAM is created and then grown. The process terminates when Ω is empty.

The algorithm groups two regions in the same GAM if they have a similar affine transformation or if there is some region with coherent intermediate affine transformation spatially located between them. In other words, the affine transformation can vary gradually from a region to the next within a GAM. Hence, a GAM can cover not only a planar, but also a curved or even a continuously deformed surface

[1]This is the only parameter, and it is set to $1/15$ of the image size.

(like bending of paper or cloth). The fact that the method doesn't prescribe a fixed neighborhood area where to grow renders it capable of grouping also spatially sparse and discontiguous subsets of correct matches.

In principle, the composition of a produced GAM might depend on the choice of its first region in step 1. However, the near-to-far growing order and the distance-based weighting make the algorithm highly order independent. In our experiments the composition of the GAMs was stable (variations of about 1%) in spite of random permutations of the input regions.

The GAM decomposition has two *fundamental properties*. First, it is unlikely for mismatches to form large GAMs (more than 4-5 matches). Mismatches have independent, widely varying, inconsistent affine transformations because they are randomly spread in the large 6D affine transformation space. The chances that N mismatches have tranformations varying slowly and gradually from one to the next quickly drop with N. On the other hand, several correct matches lying on the same surface will form a larger GAM. If a GAM is composed of many matches, these are very probably correct. Second, a GAM is most often composed of either only correct matches or only mismatches. This is because when growing a correct GAM it is unlikely for a mismatch to offer a suitable transformation. Even if this happens, the chances to add a second mismatch are again as low. As a consequence, typically mismatches are scattered over many small GAMs, while correct matches concentrate in a few larger GAMs. This brings a major advantage of organizing matches into GAMs: the number of matches in a GAM provides an indication of their probability of being correct. Moreover, the larger a GAM is, the more relevant it is, because it covers a larger part of the scene/object.

To validate the two fundamental properties, we have matched 14 image pairs, run the GAM constructor, and measured size and composition of all resulting GAMs. The total 2253 region matches have been partitioned into 1428 GAMs. 50 of them contain all 415 correct matches (*correct GAMs*), while the other GAMs have only mismatches (*incorrect GAMs*). Since the overall ratio of correct matches is only 18.4%, the statistics are relevant and truly summarize the behavior of the GAM constructor. The second property is well confirmed: 96.4% of all non-singleton GAMs are composed of either only correct matches or only mismatches (as the property trivially holds for singleton GAMs, they are omitted from this statistic). The property is also almost fulfilled by the remaining 3.6% of GAMs, as they contain all correct matches, but one (2.4%) or two (1.2%). The relation between the size of a GAM and its probability of being correct is shown in figure 2-top-right, which plots the percentage of correct GAMs of size N, for varying N. The chances that a GAM is correct quickly grow with its size and is 94% for $N > 6$.

3. Two-view filtering

This section applies GAMs to the problem of robust two-view filtering. This application is unrelated to the OR from multiple-model views, covered by the next sections.

Given a set of matches containing a large amount of mismatches, say more than 80%, we need to tell the correct matches apart. Unfortunately, the widely used RANSAC Epipolar Geometry (RANSAC-EG [10]) filter performs poorly in presence of more than 60% mismatches. Instead, we can partition the input matches into GAMs, and build on their fundamental properties. Rather than deciding on which matches to keep, we can decide on which complete GAMs to keep. GAMs are seen as the new atomic units. Moreover, since GAMs composed by more matches are more likely to be correct, we can design a powerful filter by relying more on the larger GAMs and using them to validate the smaller ones. Next, we introduce such a GAM-based filter which can handle very high amounts of mismatches (we demonstrate it up to 96%).

The algorithm starts by constructing GAMs from the input matches. Next, the GAMs are sorted according to their size, from the ones containing the most regions, to the ones containing the least. In the first iteration, a fundamental matrix is fit to all regions in the first (largest) GAM. Then, the number of inliers to this fundamental matrix are computed, among all matches within the first GAM. The first GAM is implicitly assumed correct. In the second iteration, a new fundamental matrix is fit to the first and the second largest GAMs. If there are more inliers to this new matrix than there were before, then the second GAM is deemed correct. All iterations have this general form. In the kth iteration, a fundamental matrix is fit to all GAMs previously deemed correct plus the kth one. If the number of inliers to this fundamental matrix exceeds the maximum observed so far, the kth GAM is considered correct. The inliers are computed among the matches within all GAMs previously deemed correct, plus the kth one. The algorithm iterates until it meets the first singleton GAM. At this point, all singleton GAMs which are inliers to the latest fundamental matrix are considered correct. All region matches within all GAMs deemed correct are returned as correct matches.

The power of this simple algorithm lies in the *order* in which the GAMs are inspected. In the first few iterations, the algorithm is likely to meet only correct matches, and safely builds a solid fundamental matrix. In the later iterations, this helps deciding whether smaller GAMs are correct. Besides, erroneous decisions matter less in the later iterations, as less matches are at stake. Note how the atomicity of GAMs is respected: either all regions of a GAM are accepted, or none.

The method has only two parameters: the maximal distance to the epipolar line, used to determine the inliers, and the affine dissimilarity threshold used in the GAM construc-

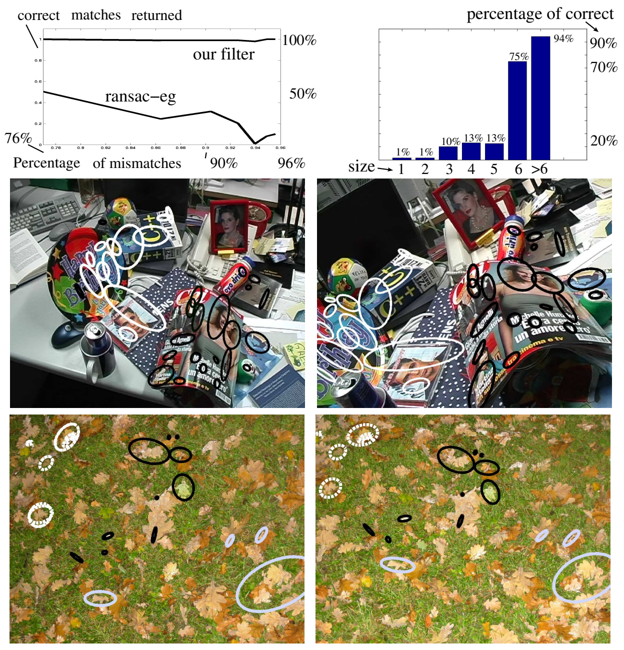

Figure 2: *Results for GAM constructor and two-view filter.*

tor. These parameters are intuitive and easy to set, in contrast to RANSAC-EG, which requires the number of iterations as a parameter. This is equivalent to providing the maximal ratio of mismatches, which is unknown a priori, and varies from case to case.

Figure 2-middle shows an example case. There are 122 initial matches, 98 of which are correct. Our algorithm groups 96 of the correct matches into the two largest GAMs (sized 59 and 37 respectively; the figure shows some matches from these two GAMs). Note how one correct GAM covers a curved surface. The mismatches are spread over 22 singleton GAMs and only one GAM sized 2. In order to test the robustness of the GAM partitioning and two-view filtering algorithms, we have added an increasing amount of randomly generated mismatches to the initial set of matches. We added from 300 to 2100 random mismatches in 7 steps, therefore raising the total percentage of mismatches from the initial 20% gradually up to 96%. At each step, we recomputed the GAMs and applied the filter. As shown in figure 2-top-left, the filter performs very well, steadily returning at least 98% of the correct matches, also when as much as 96% of mismatches contaminate the input. At the same time, the percentage of accepted mismatches stays approximately constant at 2.5%. The composition of the two largest GAMs remains identical in the first 5 steps, and changes only marginally in the last two steps. The largest incorrect GAM never contains more than 4 matches. As a comparison, the figure also shows the performance of RANSAC-EG. Even though it works very well in the initial case, returning all correct matches but one, its performance drops sharply with increasing amounts of mismatches. Already when adding 300 mismatches, only 51% of the correct matches are returned. This value continuously decreases until about 8% in the last two steps. Along all steps, an approximately constant rate of about 4.5% mismatches are accepted.

The foliage scene in figure 2-bottom illustrates a practical application of the new filter. Severe matching ambiguities arise due to the similar structures repeated many times in the images. This results in many more mismatches than in usual scenes. The initial matches set contains 25 correct matches out of 206 (12%). In these conditions, RANSAC-EG fails. However, the largest 3 GAMs are correct and contain 23 correct matches (the 3 GAMs have respectively 11, 8 and 4 matches, see figure). The mismatches are scattered in mostly singleton GAMs, with the largest incorrect GAM having only size 3. Our filter returns 24 correct matches and 5 mismatches, therefore qualitatively solving the problem.

The observed weakness of RANSAC-EG is in agreement with the independent works of [2, 1]. The experiments reconfirm the advantageous properties of GAMs: correct matches and mismatches are separated into different GAMs, while mismatches form GAMs composed of at most a few regions. These properties hold largely independently of the amount of mismatches in the input. The GAM-based filter strongly outperforms RANSAC-EG and is shown robust to very high amounts of mismatches.

4. Object modeling

Let us now turn to the main topic of the paper: how to exploit the relationships between multiple model views for recognition. This section explains how we model the object, while section 5 covers the actual recognition process.

The relationships among the model views are modeled by a dense set of *region-tracks*. Each such track is composed by the image regions of a single physical surface patch along the model views in which it is visible. The set of tracks should densely connect the model views, because they will be used during recognition in order to establish connections among GAMs matched from different model views to the test image (section 5).

This section explains how we build the model region-tracks, starting from the bare set of M unordered model images. First, dense two-view matches are produced between all pairs of model images (subsection 4.1). All pairwise sets of matches are then integrated into a single multi-view model (subsection 4.2).

4.1 Dense two-view correspondences

A dense set of region correspondences between two model views v_i and v_j is obtained by our method [4]. In summary, the method first generates a large set of unreliable, initial region matches, and then gradually *explores* the surrounding areas, trying to generate more and more matches, increasingly farther from the initial ones. The exploration process exploits the geometric transformations of existing matches to construct correspondences in v_j, for a number of overlapping circular regions, arranged on a grid completely covering the first model view v_i (*coverage regions*). This is achieved by iteratively alternating expansion phases, which construct new matching regions in v_j, with contraction phases, which remove mismatches. With each iteration, the correct matches cover more and more of the object, while the ratio of mismatches progressively decreases. The method works well in the presence of non-rigid deformations, like folding and bending [4]. Although this usually does not happen during modeling, it is an important feature at recognition time. The final outcome is large set of reliable region correspondences, densely covering the parts of the object visible in both views.

4.2 Dense multi-view correspondences

Once two-view region correspondences have been produced for all ordered pairs of model views $(v_i, v_j), i \neq j$, they can be organized into multi-view region tracks. When matching a view v_i to any of the other model views, we always use the same set of coverage regions. Therefore, each coverage region, together with the regions it matches in the other views, induces a region track. Note that if a region is matched from view v_i to view v_j, and also from view v_i to view v_k, then it is implicitly matched between v_j and v_k as well, because it will be part of the same track. These *transitive matches* actively contribute to the interview connectedness, as they often link parts of the object that are harder to match directly.
The final set of region tracks constitutes our object model.

5. Object recognition

Given a test image, the system should determine if it contains the modeled object. The first step is to match each model view of the object to the test image separately. For this purpose, the algorithm of [4] is used again. Each resulting set of region matches is then partitioned into GAMs, via the algorithm of section 2. When applied to these dense matches, the GAM decomposition is most meaningful. Each correct GAM then usually corresponds to (part of) an object facet (figure 3, only the contour of each GAM is shown).

However, at this stage, there is no guarantee that all GAMs are correct. As a result, there usually are some inconsistencies in the set of GAMs. For instance, a GAM correctly matches the head in figure 3 from model view 1 to the test image. Furthermore, there is another GAM erroneously matching the paw in model view 2 to the chest in the test image. Since the model views are interconnected by the model tracks, the model knows the correspondences among the regions on the paw in model views 1 and 2. Therefore

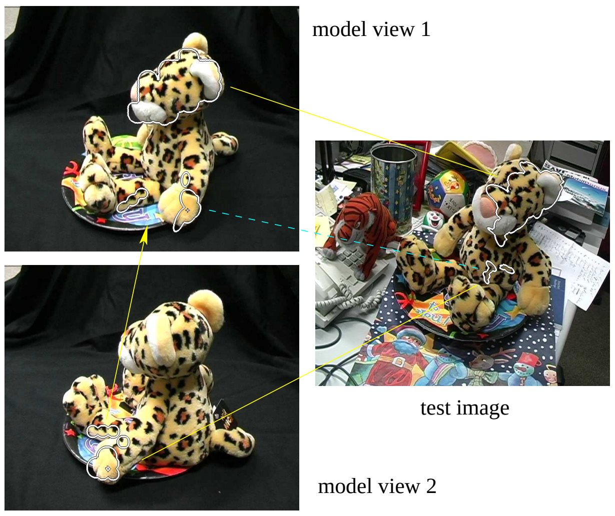

Figure 3: *The incorrect GAM on the paw is transferred from model view 2 to model view 1 (arrow).*

it considers that the second GAM matches the chest in the test image to the paw in model view 1. Now both GAMs match model view 1 to the test image, and their (geometric) inconsistency can be measured and discovered.

Just as it finds conflicting GAMs, the system can notice GAMs that are compatible. This is a good reason for considering them as more reliable and therefore to reinforce the system's belief in the presence of the object. This leads to the main advantage in evaluating GAM compatibilities: the reliability of the recognition decision is enhanced, because higher scores can be assigned in positive cases (i.e. when the object is in the test image). As a secondary advantage, incorrect GAMs can be detected and removed, thus improving the segmentation.

This section explains how to realize these ideas. But before going into the details, we sketch the overall procedure. For each pair of GAMs, we compute a compatibility score, based on the geometric consistency of their relative arrangement. In simple cases, the two GAMs are matched from the same model view and the score can be computed directly. In the more interesting cases where each GAM is from a different model view, we first have to *transfer* one of the GAMs to the model view of the other. This is made possible by the connections embedded in the model tracks. Next, the pairwise scores are integrated in a single *configuration score*. This varies as a function of the *configuration*, the subset of all GAMs which are considered correct. The process favors highly compatible subsets containing large GAMs. This is justified because larger GAMs are more likely to be correct. A genetic algorithm is used to maximize the configuration score. The maximum yields the final recognition score and reveals which GAMs are deemed incorrect. The recognition score increases in presence of compatible GAMs, thereby improving recognition performance.

The final recognition score, and the decisions to remove GAMs, are based on a *global* analysis of the situation. This considers simultaneously relationships among all pairs of GAMS, coming from all model views. It is computationally feasible because there are much less GAMs (a few tens) than region matches (hundreds to thousands). This is an advantage of reasoning on the higher perceptual grouping level offered by GAMs. The system no longer needs to consider each single region individually, but it can rely on a meaningful organization instead.
The following subsections describe the elements of the above strategy in more detail.

5.1 GAM transfer

Consider a GAM matched from a model view v_i to the test image, and another GAM matched from a different model view v_j. Before computing the compatibility score for this GAM pair, they must be put in a common model view. Only then the geometrical coherence of their relative arrangement can be evaluated.
A GAM is transferred from v_i to v_j as follows:

1. Determine the set of model regions Λ covering the same part of v_i as the GAM. This is implemented by selecting the model regions which strongly overlap (more than 70%) with the image area covered by the union of the GAM's regions. Remove from Λ all regions which are not part of a model track passing through v_j. The model can now predict the location and shape of the GAM in v_j.
2. Compute the affine transformations mapping each region of Λ from v_i to v_j.
3. Project each GAM region to v_j via the affine transformation of the nearest region of Λ. We have established a region-to-region correspondence for the GAM between the test image and model view v_j (figure 3).

When transferring a GAM, it is like making a model-based prediction. The pairwise compatibility score (next subsection) evaluates to which degree the two GAMs are consistent with this prediction. This idea is essential: in this way the system exploits the relationships among the model views, in order to conclude more than what is possible from the mere collection of all GAMs. During modeling, the system learned the structure of the object in the form of region tracks, and it brings this insight to bear at recognition time by imposing order on the GAMs.

Note that a GAM cannot be transferred if the model regions it covers in view v_i are not visible in view v_j (Λ is empty). In these cases, the compatibility score is not computed, and a neutral score is assigned instead. Note that both GAMs could still be correct, but the object parts they cover might not be visible in a single view at the same time.

5.2 Pairwise compatibility score

We evaluate here the geometric consistency of a pair of GAMs. Both GAMs are matched between the test image and a model view v_i. If at least one of the two GAMs is incorrect, we wish this measure to be low.

The compatibility score is based on the *sidedness constraint* for unordered triples of region matches [3]. The center of the first region should be on the same side of the directed line going from the center of the second region to the center of the third region, in both the model and the test image. This holds for all coplanar triples of correct matches and also for most non-coplanar ones [3]. In [3], this was first exploited as the basis for a two-view mismatch filter.

We check the constraint for all triples formed by a region from a GAM and two more regions from the other GAM. The percentage of triples respecting the constraint is our choice for the compatibility score of the GAM pair. The central idea is that if a region is picked from an incorrect GAM, we expect that most of the triples in which it takes part violate the constraint. Note that no triple is composed of regions from a single GAM. This preserves the quality of the measure when exactly one of the GAMs is correct.

The proposed score tolerates a substantial amount of non-rigid deformation. This preserves the system's capability of recognizing deformable objects. Moreover, it is insensitive to inaccurately localized region matches, because the number of triples violating the constraint grows smoothly and slowly with a region departing from its ideal location. This might happen, for instance, when a region is bridging a non-planar part of the object.

The score can penalize conflicting GAMs (e.g.: in the head-paw example above), but also assign high scores to compatible pairs of GAMs. Although the score is based on comparing region matches, it captures the compatibility of the GAMs as a whole.

5.3 Configuration score

The compatibility scores are computed for all pairs of GAMs, and are combined here in a single *configuration score*.

The compatibility scores range in $[0, 1]$. Based on a threshold t, we linearly transform the interval $[0, t]$ to $[-1, 0]$ and the interval $[t, 1]$ to $[0, 1]$. The resulting values then range in $[-1, 1]$. In all our experiments, the same threshold $t = 0.2$ splits the original range into positive and negative parts. Positive scores now indicate that two GAMs are likely to belong together, while negative ones indicate incompatibility.

We call a *configuration* C a subset of the available GAMs. What is the score of a configuration ? It should be high when containing large, mutually compatible GAMs. It should be lower in presence of incompatible ones. These two forces, pairwise corroboration and individual size, are combined into the following configuration score:

$$S(C) = \sum_{P \in C} \left(\mathrm{Size}(P) + \sum_{Q \in C \setminus P} \mathrm{Comp}(P,Q) \cdot \mathrm{Size}(Q) \right) \quad (2)$$

The number of regions in GAM P is denoted $\mathrm{Size}(P)$, while $\mathrm{Comp}(P,Q) \in [-1,1]$ are the pairwise compatibility scores. We are interested in the maximum value of $S(C)$, and in the configuration for which it occurs. The maximum value is used as recognition criteria, to decide whether the object is in the test image. Much like in section 3, more trust is given to the larger GAMs (first summation term). The second term makes the contribution of each GAM heavily dependent on its compatibility with the others, especially the larger ones. A GAM whose negative compatibilities lower S will be left out. Smaller GAM can also be part of the maximum configuration, depending on how compatible they are with the others.

An important effect of the second summation term is that the total score can be *much higher* than the mere sum of the sizes of all correct GAMs. This reflects the key idea that compatible configurations are worth more because they more reliably indicate the presence of the object. This increases the separation between scores in positive and negative cases, thus improving discriminative power.

The GAMs not selected by the best configuration are deemed incorrect and are removed. Note how this decision is based on a global analysis. Typically, several incorrect GAMs are detected thanks to their incompatibility with GAMs matched to other model views. Such a case couldn't have been discovered based on the GAM's model view alone. This is another benefit of our proposal for integrating multiple model views. In analogy with section 3, each GAM is treated as an atomic unit.

5.4 Maximization by GA

We now need to find the configuration which maximizes function (2). Unfortunately, we can't try them all out, as there are 2^n possible configurations of n GAMs.

We designed a Genetic Algorithm (GA) to find an approximation of the solution. GAs offer an elegant and flexible framework for optimizing functions of any form. In this context, we represent a configuration by a binary indicator vector I of length n. If $I(p) = 1$, the pth GAM is in the configuration. The *fitness function* $F(I)$ is defined equivalent to $S(C)$. The GA follows several steps:

1. *Initialize.* Create a random, uniformly distributed population of binary n-vectors. The size of this population is $l = \mathrm{ceil}(\sqrt{2n})^2$. Since this enforces $\sqrt{l}$ to be an integer, it simplifies the later crossover.
2. *Fitness.* Evaluate the fitness function $F(I)$ for each individual. Stop if the best individual is identical as in the previous generation (not tested the first time).
3. *Crossover.* Consider the best $\sqrt{l}$ individuals. Derive the next generation by crossing over all pairs of them. Crossing over two individuals means keeping the identical bits and randomly choosing the different bits. This amounts to producing $l - \sqrt{l}$ new individuals, and copying the current best $\sqrt{l}$.
4. *Mutation.* Each bit of each individual in the new population is switched with probability 0.1. This avoids that the algorithm explores only the part of the search space spanned by the best individuals.
5. *Iterate.* Iterate to 2.

In various experiments, this GA proved effective by approximating the true exhaustive search solution to less than 1 small GAM difference in average, on comparisons with up to $n = 20$ GAMs. It is also very time efficient, and solves cases with $n = 20$ within some seconds (exhaustive search needs more than 1 hour), and scales well, taking less than one minute for $n = 60$, a problem size for which the real optimum cannot be computed.

6. Results and conclusions

We present results for the example object of figure 3. It features a complex geometry composed by several curved surfaces. Moreover, it is covered by ambiguous texture, formed by many small variations on the same basic pattern, which challenge the matching process. The model is built from only 8 views, taken at 45 degrees intervals, all at about the same height, during a tour around the vertical axis. Figures 3 and 4-left show 4 of the views.

On the example of figure 3, the system initially produces 33 GAMs by matching each model view to the test image, via the method of [4]. Only 9 of the GAMs are correct, but 4 of them are very large (more than 60 matches) and contain the majority of the correctly matched regions. The method proposed in this paper returns 10 GAMs in the configuration with the maximal score. All of the 9 correct GAMs are included, while all but one of the 24 erroneous GAMs are detected and discarded. The final recognition score is 1770, which is three times as much as the total number of matches in the correct GAMs (596). Hence the confidence about the presence of the object is significantly boosted, compared to the system we started from [4], which just considers the total number of matches as score. Moreover, when the object is not in the test image, our approach decreases the confidence score. As combined effect, the scores assigned in the two cases are more widely separated, which leads to enhanced discriminative power. Figure 4-top-right shows the complete and accurate segmentation, as the total area covered by the 10 selected GAMs.

A challenging case is shown in figure 4-bottom-middle. The viewpoint is almost completely from above, and remarkably different from any model view. The object ap-

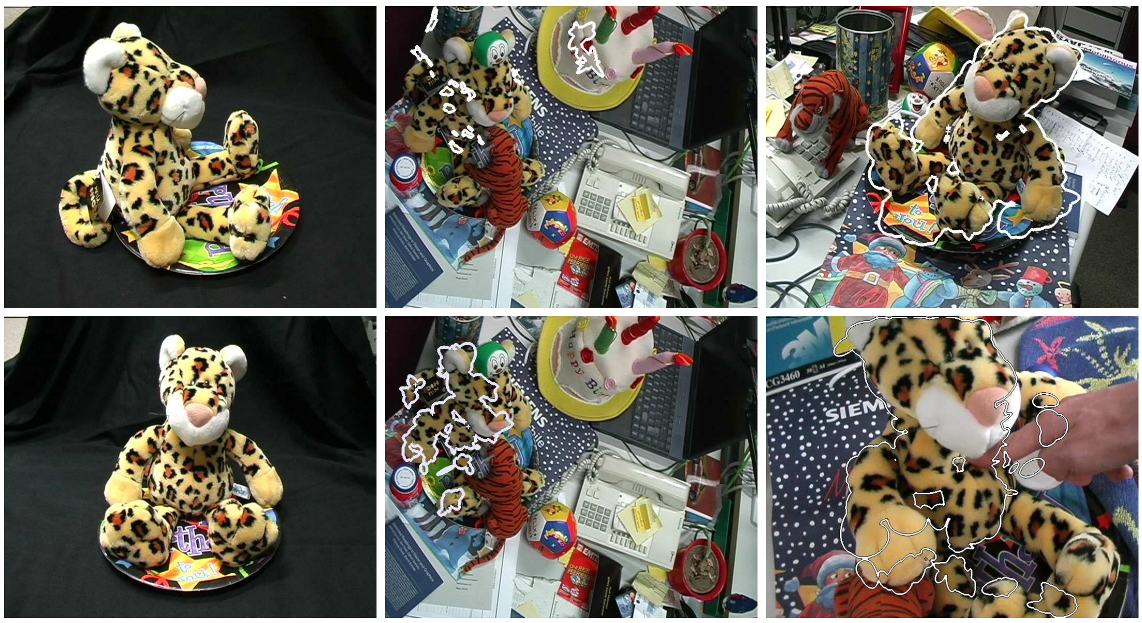
Figure 4: *Results. Left: model views. Middle: second case. Right: first and third cases.*

pears twice smaller than in the model views, and is partially occluded by a ball (head) and a Plush wildcat (front). 37 GAMs are initially produced, out of which 5 are correct. Most of the 32 wrong ones are composed by only a few matches. Our method selects all 5 correct GAMs, and 3 small incorrect ones, thereby effectively removing the large majority of them. The recognition score is 581, i.e. 2.6 times higher than the number of correct matches (216). Note the good quality of the segmentation, which includes even parts of the tail and the left paw. The system has overcome the aforementioned difficulties. Figure 4-top-middle shows some of the *removed* GAMs. The last case (figure 4-bottom-right) demonstrates recognition in presence of non-rigid deformations (raised arm, compressed chest).

Although preliminary, we believe these experiments to show that the proposed method successfully improves recognition performance by exploiting the relationships between multiple model views. The discriminative power is enhanced due to the higher scores in positive cases, and the segmentation quality improves due to the removal of spurious region matches. Multi-view integration is achieved without rigidity assumptions, and without constructing a 3D model. This success is due also to the newly introduced GAM representation. Because they are potentially valuable in several contexts of computer vision, GAMs are interesting in their own right. This was demonstrated by a powerful GAM-based two-view filter, that is largely insensitive to the percentage of mismatches. In a way, GAMs also form an alternative to the elusive concept of 'object parts', in that they offer a perceptual unit between the local features and the global object.

In the future, we plan to make the system more active. Instead of matching to all model views, we could match to the first, and then exploit the model connections to decide if and which other model view to try out. Another important extension is the support for sparsely textured objects.

References

[1] H. Chen and P. Meer, Robust Regression with Projection Based M-estimators, In *ICCV*, 2003.

[2] O. Chum, J. Matas, S. Obdrzalek, Epipolar Geometry from three correspondences In *CVWW*, 2003.

[3] V. Ferrari, T. Tuytelaars, L. van Gool, Wide-baseline Multiple-view Correspondences, *CVPR*, 2003.

[4] V. Ferrari, T. Tuytelaars, L. van Gool, Simultaneous Object Recognition and Segmentation by Image Exploration, *ECCV*, 2004.

[5] D. Lowe, Local Feature View Clustering for 3D Object Recognition, *CVPR*, 2001.

[6] J. Matas, O. Chum, M. Urban and T. Pajdla Robust Wide Baseline Stereo from Maximally Stable Extremal Regions *BMVC*, 2002.

[7] K. Mikolajczyk and C. Schmid An affine invariant interest point detector *ECCV*, 2002.

[8] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, 3D Object Modeling and Recognition Using Affine-Invariant Patches and Multi-View Spatial Constraints, *CVPR*, 2003.

[9] A. Selinger, R. C. Nelson, A Perceptual Grouping Hierarchy for Appearance-Based 3D Object Recognition, *CVIV*, 1999.

[10] P.H.S. Torr and D. W. Murray, The development and comparison of robust methods for estimating the fundamental matrix, *IJCV*, 24:3, 1997

[11] T. Tuytelaars and L. Van Gool Content-based Image Retrieval based on Local Affinely Invariant Regions *Inlt Conf. on Visual Inf. Sys.*, 1999.