Reliable and efficient pattern matching using an affine invariant metric

Michiel Hagedoorn Remco C. Veltkamp
Utrecht University, Department of Computing Science
Padualaan 14, 3584 CH Utrecht, The Netherlands
mh@cs.ruu.nl, Remco.Veltkamp@cs.ruu.nl

Abstract

In the field of pattern matching, there is a clear trade-off between effectiveness, accuracy and robustness on one hand and efficiency and simplicity on the other hand. For example, matching patterns more effectively by using a more general class of transformations usually results in a considerable increase of computational complexity. In this paper, we introduce a general pattern matching approach which will be applied to a new measure called the absolute difference. This pattern-similarity measure is affine invariant, which stands out favourably in practical use. The problem of finding a transformation mapping to the minimal absolute difference, like many pattern matching problems, has a high computational complexity. Therefore, we base our algorithm on a hierarchical subdivision of transformation space. The method applies to any affine group of transformations, allowing optimisations for rigid motion. Our implementation of the method performs well in terms of reliability and efficiency.

1 Introduction

In applications such as pose determination [15], object recognition [26], vehicle tracking [24], optical character recognition [27], stereo matching [4], content-based image retrieval [18], medical registration [25], and radiotherapy alignment [6], a major problem is finding a transformation which matches part of a pattern A to some part of another pattern B. Patterns typically consist of features extracted from industrial parts, technical drawings, sketches, hand-writing, digital pictures, and medical scans.

In this paper, general-purpose geometric pattern matching will be dealt with. The proposed method is not specialised for any specific application. Furthermore, no estimates of optimal matches will be made prior to the matching phase. The main goal is to derive a method which is general enough for matching a large variety of patterns reliably, while keeping the amount of computational effort reasonably low. No assumptions about the

input patterns will be made and no domain-specific knowledge will be incorporated in the method.

Heuristics used in most practical methods make assumptions that do not hold for more general shape matching. A well-known heuristic in pattern-matching is to align the centroids of two patterns. This eliminates translation, allowing another method to solve for the remaining degrees of freedom. In applications where patterns can have significantly large absent parts, such an approach is predestined to fail. Another heuristic is to assume that similarity is a convex function of the transformation parameters. This would allow simple hill-climbing techniques to find an optimal match. However, if rigid motions are considered, any symmetrical or nearly-symmetrical pattern will make the convexity assumption invalid, for any reasonable measure of pattern-similarity.

In contrast with many other methods, our algorithm puts only minor restrictions on transformation space. For example, when matching under affine transformations, patterns are allowed to scale to arbitrary small sizes. Since the proposed pattern similarity measure, the absolute difference, behaves well under such transformations, this poses no problems. If in a particular application, assumptions can be made about the specific regions in transformation space which can be expected to contain optimal matches, this knowledge can be put into our basic algorithm without much effort. Such restrictions of transformation-space can result in major speedups.

Geometric pattern matching, can be seen as the process of finding a mapping between two patterns in such a way that some similarity-measure is optimised. Exact pattern matching methods [3, 20, 12] decide whether or not transformations exist which map a pattern precisely onto another pattern. Unfortunately, in most practical applications measurement-errors and limited numerical precision render exact matching algorithms ineffective. Flexibility in case of perturbations and roundoff errors can be achieved by considering point-set correspondences. Correspondence-based pattern matching methods include minimum weight matching, uniform matching, minimum deviation matching and bottleneck matching [2, 1]. A major disadvantage of these methods is that, in principle, only point sets having equal sizes can be matched. Much work has been done on matching point set patterns without establishing correspondences by means of the Hausdorff distance [12, 11, 16, 9, 8, 22, 19]. Although robust against perturbations, the Hausdorff distance is very sensitive to occlusion and absent or outlying parts. The partial Hausdorff distance [7, 21] is a pattern-similarity measure which overcomes this drawback by matching only parts of patterns. As a result, the partial Hausdorff distance is not a metric. More seriously, the distance is dependent on a threshold parameter specifying the minimal fraction of matching points, which must be known a priori. Alt [14] proposes the symmetric difference which defines a metric, and if used for minimisation, deals well with outlying or absent parts in shapes. However, the symmet-

ric difference and the Hausdorff distance variants are not invariant under affine transformations, and scaling in particular. This results in unwanted behaviour of these similarity measures under scaling and affine transformations, making restrictions on transformation space necessary.

In this paper, a variation on the symmetric difference, called the absolute difference, is proposed. Like the symmetric difference, this measure is robust against perturbation, occlusion and outlying or absent parts. In contrast with the partial Hausdorff distance, the absolute difference is a metric and does not depend on a parameter. Unlike the Hausdorff distance variants and the symmetric difference, the absolute difference is affine-invariant, allowing it to be used for minimisation under any affine transformation group without the need of restricting the transformation space.

The complexity of pattern matching under both the Hausdorff distance and the symmetric difference correlates with the dimension of the transformation space under which these measures are minimised. Rucklidge [21] proved lower bounds for the number of continua in transformation space for which the directed Hausdorff distance between point sets is minimal. For planar point sets of size n, the number of minimal regions under translation, rigid motion, translation and scaling, and affine transformation are $\Omega(n^3)$, $\Omega(n^5)$, $\Omega(n^7)$ and $\Omega(n^9)$, respectively. These constructions can be modified to work for the symmetric difference and the absolute difference as well.

The pattern matching method proposed in this paper avoids explicit consideration of all minima by splitting up transformation space in a top down manner. This approach will be most efficient if optimal matches are concentrated within a limited number of small regions in transformation space. Under such circumstances our algorithm will eliminate large subsets of transformation space at an early stage. In [10], we applied our hierarchical minimisation-approach to the partial Hausdorff distance. In the current paper, an algorithm for matching using the absolute difference will be developed.

The new algorithm can be used to approximate the minimum absolute difference within any desired accuracy, reporting a near-optimal transformation in the process. The method is general: it works for points of arbitrary dimension, and various classes of transformations, including translations, scalings combined with translations, rigid transformations, similarity transformations, area-preserving transformations, and affine transformations in general.

2 Pattern and shape similarity

A pattern is a set or a function which is used to model a number of features. A shape can be seen as a pattern modulo the action of a group. For example, consider a planar point set A and a non-trivial translate B of A. Under the

group of translations, A and B have equal shapes, while seen as patterns A and B are different. A transformation group acting on a set of patterns defines a collection of shapes. The shape of a pattern is defined as the set of all images of that pattern under transformation. A space of shapes is obtained by defining a metric on shapes.

Section 2.1 defines certain topological transformation groups which can be used to transform patterns. The construction of metric spaces of patterns and shapes is discussed in Section 2.2. Section 2.3 deals with the use of metrics for measuring similarity of patterns and shape. Appendix A recalls the basic definitions of topological transformation groups and metric spaces.

2.1 Topological transformation groups

Distinct topological transformation groups will be used for modelling physical phenomena like object-movement and viewpoint-change. For example, the group of rigid motions acting on subsets of $\mathbb{R}^2$ can be used to model the freedom of movement of flat objects on a table. The affine group acting on sets in $\mathbb{R}^2$ is a good approximation for a change of camera-viewpoint under weak perspective. The group of homeomorphisms acting on $\mathbb{R}^3$ can be used to model three-dimensional tissue-deformations.

The set of all homeomorphisms on a fixed space is denoted by $\mathbf{Hom}$. Homeomorphisms are continuous invertible mappings having a continuous inverse. Every topological transformation group acting on a space is a subgroup of the group of homeomorphisms of that same space. For example, the general linear group $\mathbf{GL}^d$ consists of all non-singular $d \times d$-matrices. A more restricted transformation group is $\mathbf{SL}^d$, the group of $d \times d$-matrices having a unit determinant. The orthogonal group, a subgroup of $\mathbf{GL}^d$, is the collection $\mathbf{O}^d$ of all non-singular matrices A for which $AA^t = I$. A subgroup of both $\mathbf{SL}^d$ and $\mathbf{O}^d$, the special orthogonal group, is defined by $\mathbf{SO}^d = \mathbf{O}^d \cap \mathbf{SL}^d$. The non-linear group of d-dimensional translations is denoted by $\mathbf{T}^d$.

The composition $G \cdot H$ of two groups G and H consists of all gh, where $g \in G$ and $h \in H$. The affine transformations $\mathbf{A}^d$ and the isometries $\mathbf{M}^d$ are the compositions $\mathbf{T}^d \cdot \mathbf{GL}^d$ and $\mathbf{T}^d \cdot \mathbf{SO}^d$, respectively. and the as $\mathbf{T}^d \cdot \mathbf{SO}^d$. The special affine transformations and the rigid transformations are defined by $\mathbf{SA}^d = \mathbf{T}^d \cdot \mathbf{SL}^d$ and $\mathbf{SM}^d = \mathbf{T}^d \cdot \mathbf{SO}^d$, respectively.

Each group has specific invariants. For example, the ratio of areas and line parallelism are affine invariants. The special affine transformations preserve area. Isometries and rigid transformations preserve both length and area.

2.2 The shape of a pattern

Shapes are defined as patterns modulo the action of a transformation group. Let G be a topological transformation group acting on a set of patterns P. Then each pattern $A \in P$ determines a unique shape GA consisting of the set of all images of A under G, that is, $GA = \{gA \mid g \in G\}$. The group G defines an equivalence relation $\sim_G$ on P, where $A \sim_G B$ if and only if there exists a transformation $g \in G$ such that $gA = B$, for $A, B \in P$. Therefore, a shape GA of a pattern $A \in P$ can be seen as the equivalence class in P under $\sim_G$ containing A. The class of shapes P/G determined by G and P is the set of all these equivalence classes $P/G = \{GA \mid A \in P\}$. This means that patterns A and B have the same shape when a transformation $g \in G$ exists such that $gA = B$.

The set of patterns P and the collection of corresponding shapes P/G will be treated as metric spaces. Choosing a metric λ on P results in a metric space of shapes (P, λ). The metric λ can be used to model pattern-similarity. That is, the choice of pattern-metric determines how similar patterns are.

A metric λ on the set of patterns P is said to be G-invariant if $\lambda(gA, gB) = \lambda(A, B)$ for all $g \in G$ and $A, B \in P$. The following theorem shows how to construct a metric shape space $(P/G, \mu)$ from a metric pattern space (P, λ).

Theorem 2.1 *Let G be a group acting on a metric space (P, λ). Suppose λ is G-invariant. Then $\mu(GA, GB) = \min_{g \in G} \lambda(gA, B)$ defines a metric space $(P/G, \mu)$.*

A less general presentation of this result was given by Rucklidge in [21].

If λ provides a good measure of pattern-perturbation, the shape-metric induced by Theorem 2.1 gives a measure μ that is reliable when pattern-perturbations occur. The mapping $\pi : A \mapsto A/G$, which computes the shape GA corresponding to a pattern $A \in P$, is continuous with respect to the metric spaces (P, λ) and $(P/G, \mu)$. The continuity of π implies that introducing small changes in patterns $A, B \in P$ will have a small impact on the distance $\mu(GA, GB)$ between the shapes GA and GB.

From here on, the symbol λ always denotes a metric on the set of patterns P, while μ denotes a metric on the collection of shapes P/G. The next section discusses criteria for good pattern similarity measures.

2.3 Measuring similarity of patterns and shapes

In practical applications, patterns inevitably suffer from various defects caused by aberration, scanning inaccuracies, limited graphical resolution, roundoff-errors or unreliable edge detection. A reliable pattern matcher must be based on a similarity measure that behaves well in case such defects occur. Therefore, a pattern-similarity measure $\lambda : P \times P \to \mathbb{R}$ must satisfy the following conditions:

1. The function λ must be a metric. Identical shapes should have distance zero. No negative distances should be allowed. The notion of similarity should be symmetrical. If the similarity between GA and GB and between GB and GC are high, the similarity between GA and GC should be high.
2. Small deformations in patterns A and B must result in a small change in the pattern-distance $\lambda(A, B)$.
3. The distance λ should be low if two patterns have large, highly similar parts. The distance between two patterns with large similar parts should on the average be significantly lower than the distance between any two arbitrary patterns. This means that λ must be capable of detecting similar patterns even if parts in one pattern may be absent from the other or lying outside of it.

Examples of pattern-similarity measures are the the uniform metric [23] the Fréchet distance [13], the Hausdorff distance [16], the partial Hausdorff distance [21, 10], and the symmetric difference [14]. In exact pattern matching algorithms, the discrete metric, defined by $A \neq B \Rightarrow \lambda(A, B) = 1$, is used implicitly.

Each pattern-similarity measure, has a specific pattern space P on which it is defined. Many similarity values are defined on subsets of $\mathbb{R}^d$. For example, the uniform metric is defined on x-monotone curves. The Fréchet distance measures the similarity of curves. The Hausdorff distance and its variations apply to compact sets in $\mathbb{R}^d$. The symmetric difference is well-defined on solid sets, which are compact sets that are equal to the closure of their interior.

The absolute difference, which will be defined below, behaves similar to the symmetric difference in the case of rigid transformations. The previous similarity measures were defined on a set of patterns P containing subsets of some space X. The absolute difference, on the other hand, uses a class of patterns P_a in which each pattern $A \in P_a$ is a function $A : X \to \mathbb{R}$.

Definition 2.2 *The collection of patterns P_a consists of all functions $A : V \to \mathbb{R}$ which can be written as*

$$A(x) = \begin{cases} h(a_i) & \text{if } x \in D(a_i) \\ 0 & \text{otherwise} \end{cases}$$

for a finite set $\{a_i\}$ determining disjoint solids $D(a_i)$ and reals $h(a_i)$.

Functions in P_a have constant value over a finite number of solid sets. The elements of P_a will be called solid functions.

Definition 2.3 *The absolute difference λ_a is defined by*

$$\lambda_a(A, B) = \int_{x \in X} |A(x) - B(x)| \, dx$$

for $A, B \in P_a$.

The symmetric difference λ_s on solid sets in X can be written in terms of λ_a by $\lambda_s(A, B) = \lambda_a(\tilde{A}, \tilde{B})$, where $\tilde{A}(x) = 1$ if $x \in A$ and 0 otherwise.

The area ratio $\mathrm{ar}(g)$ is defined as the absolute value of the determinant of the linear component of an affine transformation g. This function allows the following definition of an action of the affine group $\mathbf{A}^d$ on P_a.

Definition 2.4 *Let $g \in G$ and $A \in P_a$. Then gA is the solid function defined by*

$$gA(gx) = \mathrm{ar}(g)^{-1} A(x)$$

for $x \in X$.

This action preserves the area between A and the zero function. The following lemma shows that (P_a, λ_a) is a metric space.

Lemma 2.5 *The absolute difference is a metric.*

Proof: The metric properties (i)–(iv) enumerated in Appendix A must hold. Properties (i) and (iii) follow directly from the definition.

First, property (ii) will be be dealt with. It must be shown that $\lambda_a(A, B) = 0$ implies $A = B$. The reverse implication is trivial. Assume that the solid functions $A, B \in P_a$ are determined in the sense of Definition 2.2 by $\{a_i\}$ and $\{b_j\}$ respectively.

Let $\{U_k\}$ consist of all non-empty intersections $D(a_i) \cap D(b_j)$. Clearly, $\{U_k\}$ is a collection of solid sets on which both A and B have constant value. The absolute difference λ_a can be written as a sum of integrals over non-negative constant functions:

$$\lambda_a(A, B) = \int\limits_{x \notin \cup_i D(a_i)} |B(x)|\, dx + \int\limits_{x \notin \cup_j D(b_j)} |A(x)|\, dx + \sum_k \int\limits_{x \in U_k} |A(x) - B(x)|\, dx$$

Assume $A(y) \neq B(y)$ for some $y \in X$. Because $|A(y) - B(y)| > 0$, the point y must lie in $\cup_i D(a_i)$ or $\cup_j D(b_j)$. Suppose $y \in U_k$. Then $|A(x) - B(x)| > 0$ for $x \in U_k$, implying $\int_{x \in U_k} |A(x) - B(x)|\, dx > 0$. Suppose $y \in D(a_i) - \cup_j D(a_j)$. Then y must lie in the non-empty difference of two solid sets. Therefore, there must exist a non-empty solid set S such that $y \in S \subseteq D(a_i) - \cup_j D(b_j)$. This implies $\int_{x \notin \cup_j D(b_j)} |B(x)|\, dx > 0$. The remaining case $x \in D(b_j) - \cup_i D(a_i)$ is symmetrical.

Metric property (iv) remains. Integrating the inequality $|A(x) - C(x)| \leq |A(x) - B(x)| + |C(x) - B(x)|$ gives the triangle inequality.

□

The next Lemma shows that, unlike the symmetric difference, the absolute difference is affine-invariant. Therefore, the minimum absolute difference under affine transformation is a well-defined shape-distance. This

distance (λ)	patterns (P)	G	metric	robust	absent
discrete	arbitrary subsets	$\mathbf{Hom}$	yes	no	no
uniform metric	planar curves	$\mathbf{T}^2$	yes	yes	no
Fréchet	curves	$\mathbf{M}^d$	yes	yes	no
Hausdorff	compact subsets	$\mathbf{M}^d$	yes	yes	no
partial Hausdorff	compact subsets	$\mathbf{M}^d$	no	yes	yes
symmetric difference	solid subsets	$\mathbf{SA}^d$	yes	yes	yes
absolute difference	solid functions	$\mathbf{A}^d$	yes	yes	yes

Table 1: Comparing properties of pattern-similarity measures.

means that, unlike the Hausdorff distance, the absolute difference behaves well under affine minimisation, making ad-hoc restrictions on transformation space unnecessary.

Lemma 2.6 *The absolute difference is invariant under the group of affine transformations.*

Proof: Invariance under the group of translations $\mathbf{T}^d$ is trivial. Here, $\mathbf{GL}^d$-invariance will be shown.

Write λ_a as a sum of integrals as in Lemma 2.5. Let D be some solid U_i, a set $D(a_i) - \cup_j D(b_i)$ or a set $D(b_j) - \cup_i D(a_i)$. Suppose $|A(x) - B(x)| = h$, for all $x \in D$. Then,

$$\begin{aligned}
\int_{x\in D} |A(x) - B(x)|\, dx &= \int_{x\in D} h\, dx \\
&= \operatorname{ar}(g)^{-1} \int_{y\in gD} h\, dx \\
&= \int_{y\in gD} |\operatorname{ar}(g)^{-1} A(y) - \operatorname{ar}(g)^{-1} B(y)|\, dx \\
&= \int_{y\in gD} |gA(y) - gB(y)|\, dx
\end{aligned}$$

for $g \in \mathbf{GL}^d$. Thus each term of the sum is invariant under the action of $\mathbf{GL}^d$. □

Table 1 gives an overview of the pattern-similarity measures discussed in this section. The first column contains the name of each measure. The collection of patterns for which a measure is defined is given in the second column. The third column shows the topological transformation group for which a similarity measure is invariant. The last three columns in Table 1 indicate if the various similarity measures satisfy the metric properties (1), are robust in case of pattern-perturbation (2) and deal well with outlying or absent parts (3).

Table 1 shows that both the symmetric difference and the absolute difference satisfy each of the properties 1, 2 and 3. However, the absolute difference is invariant under all affine transformations, whereas the invariance of the symmetric difference is restricted to area-preserving affine transformations.

3 Finding an optimal matching

This section explains an hierarchical pattern matching method. The procedure can be applied to any topological transformation group having a finite-dimensional real representation. The affine group and its subgroups are important examples of such groups. The general method can be applied to the variations on the Hausdorff distance, the symmetric difference, and the absolute difference.

3.1 The basic algorithm

The problem is finding a transformation $g \in G$ that minimises $\lambda(gA, B)$. In the process, the minimal value of λ must be computed. In Rucklidge's approach [21], a multi-resolution subdivision of transformation space is made in order to compute matches. Likewise, the technique discussed in this section splits up transformation space recursively. For sets of transformations, a lower bound l for the minimal value of λ for the transformed pattern gA and the pattern B is computed. In each pass of the algorithm, the set of transformations with the lowest value of l is selected and split up into smaller sets of transformations. The basic idea is that if, at some stage of the algorithm, a small set of transformations C has the lowest lower bound l yet encountered, all transformations in C map A nearly optimal to B under λ.

Let G be a transformation group acting on a metric pattern-space (P, λ). Given a pattern $A \in P$, each set of transformations $C \subseteq G$ has a corresponding set of patterns $CA \subseteq P$. The most important component of algorithm is the lower bound l for which $l(C, A, B) \leq \lambda(A', B)$ for $A' \in CA$. The bound l is only useful if it converges to the actual value of λ for sets of transformations shrinking to any single fixed transformation.

At this level, the representation of the topological transformation group G becomes significant. A representation of a group G is a continuous surjective function $\varphi : \mathbb{R}^k \to G$, mapping a finite dimensional vector space onto the set of transformations in G. Finite-dimensional representations exist for the group of affine transformations and its subgroups.

Only rectangular regions R in representation space $\mathbb{R}^k$, called cells, will be considered. Let l be a lower bound for λ, that is, $l(R, A, B) \leq \lambda(\varphi r A, B)$, for all $r \in R$. Suppose that for each $\epsilon > 0$, there exists a $\delta > 0$ such that $\mathrm{diam}(R) < \delta$ implies $\lambda(\varphi r A, B) - l(R, A, B) < \epsilon$, for $r \in R$. In this case,

FIND-MINIMAL-TRANSFORMATION(A, B, ϵ)

```
1  Compute cells {R_i} that cover the global minimum.
2  for each cell R_i
3  do Insert R_i in a priority queue Q with key l(R_i, A, B).
4  Extract the minimal cell R from Q.
5  while d(R, A, B) > ε
6  do Split R equally on its largest axis into R' and R'' .
7       Insert R' in Q with key l(R', A, B).
8       Insert R'' in Q with key l(R'', A, B).
9       Extract the minimal cell R from Q.
10 return some φr ∈ φR and λ(φr(A), B)
```

Algorithm 1: Finding a minimal transformation.

l is said to converge to λ. Since only finite-dimensional representations are considered, cells R can be split up into disjoint cells R' and R'', until a cell containing a transformation within a fixed of the global minimum is found.

Let d be a function that bounds the difference between $l(R, A, B)$ and $\lambda(\varphi r A, B)$. Algorithm 1 computes an optimising transformation within any accuracy $\epsilon > 0$ using the functions l and d. The algorithm starts with a number of initial cells which cover the global minimum. The rectangular cells are being split up repeatedly until a sufficiently accurate approximation of the global minimum is found. The next theorem states the correctness of Algorithm 1.

Theorem 3.1 *Let $A, B \in P$ and let $R \subseteq \mathbb{R}^k$. Suppose*

$$\lambda(\varphi r A, B) - l(R, A, B) < d(R, A, B),$$

for all $r \in R$, where d converges to zero as $diam(R)$ goes to zero. Then Algorithm 1 computes a transformation $g_m \in G$ which brings the distance value within any constant $\epsilon > 0$ of the global minimum value:

$$\lambda(g_m(A), B) - \min_{g \in G} \lambda(g(A), B) < \epsilon.$$

3.2 Lower bounds for pattern-metrics

In this section, a theorem concerning lower bounds for pattern-metrics will be derived. Such lower bounds can be used in Algorithm 1. After that, special instances of this theorem for the absolute difference λ_a, the Hausdorff distance λ_h and the symmetric difference λ_s will be discussed.

Many pattern-metrics λ can be decomposed as $\lambda(A, B) = \kappa(A, B) \sqcap \kappa(B, A)$, where $\sqcap$ denotes some monotone function. A binary function $\sqcap : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is called monotone if it is increasing in both arguments.

Theorem 3.2 indicates how one finds a lower bound for pattern-metrics λ which can be decomposed monotonically in two one-way pattern-distances κ.

Theorem 3.2 *Let G be a transformation group acting on a metric space (P, λ) with λ invariant under G. Furthermore, let $\lambda(A, B) = \kappa(A, B) \sqcap \kappa(B, A)$, where $\sqcap$ is monotone. Suppose τ satisfies $\kappa(\tau(C, A), B) \leq \kappa(CA, B)$. Then*

$$\lambda(CA, B) \geq \kappa(\tau(C, A), B) \sqcap \kappa(\tau(C^{-1}, B), A)$$

where $C^{-1} = \{c^{-1} \mid c \in C\}$.

Proof: The G-invariance of λ and the monotonicity of $\sqcap$ allow the following derivation:

$$\begin{aligned} \min \lambda(CA, B) &= \min\{\kappa(cA, B) \sqcap \kappa(B, cA) \mid c \in C\} \\ &= \min\{\kappa(cA, B) \sqcap \kappa(c^{-1}B, A) \mid c \in C\} \\ &\geq \min \kappa(CA, B) \sqcap \min \kappa(C^{-1}B, A) \\ &\geq \kappa(\tau(C, A), B) \sqcap \kappa(\tau(C^{-1}, B), A). \end{aligned}$$

□

Theorem 3.2 assumes that a pattern-metric λ can be written in terms of a one-way distance κ. Theorem 3.2 shows that a bound for κ can be found by considering special patterns, found using a function τ, which assume a distance lower than a collection of transformations of a single pattern. Two of such special patterns can be substituted for the individual transformed patterns in the definition of λ resulting in a lower bound.

Theorem 3.2 has a general character. It can be applied to the absolute difference, the Hausdorff distance and the symmetric difference, resulting in Corollary 3.3, 3.4 and 3.5, respectively.

Corollary 3.3 *Let $\kappa_a(A, B) = \int_{A(x)<B(x)} B(x) - A(x)\, dx$ and let $\tau_a(C, A)(x) \geq \sup_{c \in C} \mathrm{ar}(c) A(c^{-1}x)$, for $C \subseteq \mathbf{A}^d$ and solid functions A and B defined on $\mathbb{R}^d$. Then*

$$\lambda_a(CA, B) \geq \kappa_a(\tau_a(C, A), B) + \kappa_a(\tau_a(C^{-1}, B), A)$$

where $\lambda_a(A, B) = \kappa_a(A, B) + \kappa_a(B, A)$.

Proof: By Lemma 2.5 and Lemma 2.6, the absolute difference λ_a is an affine-invariant metric. Applying Theorem 3.2, gives the result. □

Corollary 3.4 *Let $\kappa_h(A, B) = \max_{b \in B} \min_{a \in A} \|b - a\|$ and let $\tau_h(C, A) \supseteq \cup CA$ be compact, for $C \subseteq \mathbf{M}^d$ and compact $A, B \subseteq \mathbb{R}^d$. Then*

$$\lambda_h(CA, B) \geq \max(\kappa_h(\tau_h(C, A), B), \kappa_h(\tau_h(C^{-1}, B), A))$$

where $\lambda_h(A, B) = \max(\kappa_h(A, B), \kappa_h(B, A))$.

Corollary 3.5 *Let* $\kappa_s(A,B) = area(B-A)$ *and let* $\tau_s(C,A) \supseteq \cup CA$ *be solid, for* $C \subseteq \mathbf{SA}^d$ *and any solid* $A, B \subseteq \mathbb{R}^d$. *Then*

$$\lambda_s(CA,B) \geq \kappa_s(\tau_s(C,A),B) + \kappa_s(\tau_s(C^{-1},B),A)$$

where $\kappa(A,B) = area(B-A)$.

Corollaries 3.3, 3.4 and 3.5 can be used to find functions l which can be used in Algorithm 1. Below, Corollary 3.3 will be applied in deriving a lower bound for the absolute difference under a cell of transformations.

3.3 Bounding the absolute difference

Algorithm 1 uses a lower bound for the pattern-distance under a set of transformations. In this section, a lower bound will be found specifically for the absolute difference λ_a under the group of affine transformations. This achieved by making use of Corollary 3.3.

To arrive at an explicit definition of a lower bound function, used in Algorithm 1, a choice of representation must be made. For representing the affine group $\mathbf{A}^d$, $d \times d$-matrices, which are vectors in $\mathbb{R}^{d^2}$, are used for representing the general linear group $\mathbf{GL}^d$, while vectors in $\mathbb{R}^d$ are used to represent translations in $\mathbf{T}^d$. Thus $d(d+1)$-dimensional vectors can be used to represent $\mathbf{A}^d$.

A traced volume is a set which contains the union of images of a solid under a set of transformations. Using the current choice of affine representation, a function V which determines a the traced volume given a rectangular solid S and a cell R will be defined. Let $R = \{r \mid \underline{r}_{ij} \leq r_{ij} \leq \overline{r}_{ij}\}$ be a block of $d \times d$-matrices and let $S = \{s \mid \underline{s}_i \leq s_i \leq \overline{s}_i\}$ be a block in $\mathbb{R}^d$. A traced volume under linear transformation of S under R is defined by the axis-parallel block $L(R,S) = \{x \mid \min t_i \leq x_i \leq \max t_i\}$, where $t_i = \{\underline{r}_{ij}\underline{s}_i, \underline{r}_{ij}\overline{s}_i, \overline{r}_{ij}\underline{s}_i, \overline{r}_{ij}\overline{s}_i\}$. The traced volume of a block S under a block of translations $R = \{\underline{r}_i \leq r_i \leq \overline{r}_i\} \subset \mathbb{R}^d$ can be expressed as $T(R,S) = \{x \mid \underline{r}_i + \underline{s}_i \leq x_i \leq \overline{r}_i + \overline{s}_i\}$. The functions L and T determine traced volumes under general linear transformation and translation, respectively. A function V computing the traced volume under affine transformation can be found by composition. Let $R = R' \times R''$ be a block of affine representations, where R' and R'' are blocks of translations and linear transformations, respectively. The traced-volume function V is now defined by $V(R,S) = T(R', L(R'', S))$.

In addition to traced volumes, a function q which bounds the area ratio $\mathrm{ar}(c)$ over transformations $c \in C$ corresponding to a cell is needed. From here on, the symbol R always denotes a cell, while $C = \varphi R$ denotes the corresponding set of transformations. The function q must satisfy $q(R) \geq \max_{c \in C} \mathrm{ar}(c)^{-1}$. Choosing a converging function q that satisfies this

property is straightforward. In the two-dimensional case the inequality can be put into an equality quite easy.

The traced volumes obtained under affine transformations are useful for finding a function τ_a which satisfies the conditions of Corollary 3.3. Below, a function τ_a which determines a special pattern $\tau_a(C, A)$ for the set transformations C and the pattern A is defined. In accordance with Corollary 3.3, τ_a is chosen such that the one-way distance κ between the special pattern $\tau_a(C, A)$ and B bounds $\kappa(cA, B)$ for all $c \in C$. This can be accomplished by taking $\tau(C, A)$ to be an upper bound for the upper envelope of all cA for $c \in C$. Thus τ_a can be defined as:

$$\tau_a(C, A)(x) = q(R) \max\{A(y) \mid x \in V(R, y)\}. \tag{1}$$

Note that $\tau_a(C, A)$ may not be a pattern in the sense of Definition 2.2. Since the value of κ is well-defined for such $\tau_a(C, A)$, this poses no problem. Using τ_a a lower bound k for the one-way absolute difference κ over a cell R can be derived:

$$k(C, A, B) = \kappa_a(\tau_a(C, A), B). \tag{2}$$

Applying Corollary 3.3 gives a lower bound l defined by

$$l(C, A, B) = k(C, A, B) + k(C^{-1}, B, A). \tag{3}$$

In the next section, algorithms for computing $l(R, A, B)$ efficiently, for varying R and fixed $A, B \in P_a$, will be discussed.

3.4 Efficient lower bound computation

An issue yet uncovered is the efficient computation of the lower bound l defined in Equation 1, 2 and 3. In this section, the discussion will be restricted to the computation of the one-way bound k defined in Equation 2. For that purpose an explicit definition of τ_a, presented in Equation 1, is desirable. Assume that patterns A and B are represented in the sense of Definition 2.2 by finite sets $\{a_i\}$ and $\{b_i\}$ respectively. The function $\tau_a(C, A)$ can be written in terms of a set of functions. Each a_i determines a function τ_a^i defined by:

$$\tau_a^i(x) = \begin{cases} q(R)\, h(a_i) & \text{if } x \in V(R, D(a_i)) \\ 0 & \text{otherwise} \end{cases}.$$

A more explicit version of Definition 1 is obtained by choosing the upper envelope of such functions:

$$\tau_a(C, A)(x) = \max_i \tau_a^i(C, A)$$

At this stage an obvious method for computing the one-way bound $\kappa(\tau_a(C, A), B)$ comes into sight. First, the upper envelope of the collection

```
BUILD-TREE(B)
 1  Let D[n] be a bounding box for each D(b_i).
 2  Let h[n] be the maximum of all h(b_i).
 3  if B is determined by only one b_i
 4     then return leaf n
 5     else  Let x be the largest axis of D[n].
 6           Let b_1, ..., b_t be sorted in the direction of x.
 7           Split b_1, ..., b_t on its median into {b'_i} and {b''_j}.
 8           c_1[n] ← BUILD-TREE(B')
 9           c_2[n] ← BUILD-TREE(B'')
10           return internal node n
```

Algorithm 2: Building an augmented kd-tree.

of functions $\tau_a^i(C, A)$ can be computed, resulting in the function $\tau_a(C, A)$. Then, during computation of the lower envelope of $\tau_a(C, A)$ and B, the one-way bound k can be evaluated. However, explicit upper bound computations are not essential for the computation of k. Therefore, a different approach is taken.

Here, an algorithm will be derived specifically for pattern-functions B which can be written as a finite set of disjoint rectangles $D(b_i)$ with associated function-values $h(b_i)$. Assume that these rectangles have some order for each axis of the coordinate system separately. The basic data-structure used in the algorithm is the kd-tree [17]. An augmented kd-tree containing records providing information which can be used to speed up the computation of the bound is used.

For a pattern B represented by $\{b_i\}$, a balanced kd-tree is built on the set of regions $D(b_i)$. Each node n in the kd-tree has children $c_1[n]$ and $c_2[n]$. Each bounding box $D[n]$ in the tree contains the bounding boxes of the child nodes $D[c_1[n]]$ and $D[c_2[n]]$. In addition, each node n has a field $h[n]$ which equals the maximal value of all $h(a_i)$ over all leaves in the subtree. Algorithm 2 gives pseudo-code for building such a kd-tree.

The computation of $k(C, A, B)$ is performed in two stages. First, the kd-tree t_B, built on B, is labelled using each of the traced volumes $V(R, D(a_i))$ and the values $q(R)\, h(a_i)$. This is done by Algorithm 3. After that, Algorithm 4 computes the actual bound for the one-way absolute difference using the labelled tree.

Algorithm 3, the tree-labelling procedure, works as follows. For each rectangle $D(a_i)$, a function τ_a^i can be defined in terms of the traced volume $V(R, D(a_i))$ and the value $q(R)\, h(a_i)$. These values are passed to the root of the kd-tree. If the sub-pattern represented by a subtree rooted at n lies completely below τ_a^i, a flag $o[n]$ is set. In addition, each node n has a field $m[n]$ which stores the minimum over all leaves l in the subtree, of the

```
LABEL-TREE(n, D, h)
 1  if ¬o[n] ∧ D[n] ∩ D ≠ ∅
 2    then if D[n] ⊆ D ∧ h[n] ≤ h
 3           then o[n] ← true
 4           else  if m[n] < h
 5                   then if n is leaf
 6                          then m[n] ← min(h[n], h)
 7                          else  LABEL-TREE(c1[n], D, h)
 8                                LABEL-TREE(c2[n], D, h)
 9                                m[n] ← min(m[c1[n]], m[c2[n]])
```

Algorithm 3: Labelling the kd-tree.

```
EVAL-TREE(n)
1  if o[n]
2    then return 0
3    else  if n is a leaf
4           then return (h[n] − m[n]) · area(D[n])
5           else  return EVAL(c1[n]) + EVAL(c2[n])
```

Algorithm 4: Evaluating a labelled tree.

maximum value of any previously processed τ_a^j having value lower than $h[l]$, for any $x \in D[l]$.

Algorithm 4 is used to evaluate the labelled tree resulting in the desired value of k. It does a recursive traversal of the kd-tree built on B starting with the root. In the process it accumulates the one-way differences between $\tau_a(C, A)$ and B over all subregions in the kd-tree. If a node is encountered for which $o[n]$ is set, it is known that the corresponding part of the pattern B lies completely underneath $\tau_a(C, A)$. In that case, a value of zero can be reported for the subtree as a whole. If a leaf n is encountered, the difference in height of $\tau_a(C, A)$ and B restricted to the domain $D[n]$ is added to the sum.

The labelling algorithm does not compute k exactly. In case a traced volume $V(R, D(a_i))$ intersects the block $D[l]$ corresponding with a leaf l in a tree, but does not contain it, it is treated as if it fully contains $D(a_i)$. This reduces the complexity of the algorithm considerably. The value computed by Algorithm 4, will in some cases be lower than the actual value of k. In the experiments, relatively small cubes a_i for which this difference is insignificant will be used.

Algorithm 5 illustrates the general procedure for computing $k(R, A, B)$, for cells R, patterns B and a kd-trees t_B. The kd-tree can be computed in

```
BOUND(R, A, t_B)
 1  for each node n in t
 2  do m[n] ← 0
 3       o[n] ← false
 4  for all a_i
 5  do LABEL-TREE(root[t_B], T(R, D(a_i)), q(R) · h(a_i))
 6  return EVAL-TREE(root[t_B])
```

Algorithm 5: Bounding the one-way absolute difference over a cell.

a pre-processing stage, previous to the execution of Algorithm 1, and can be used each time a lower bound has to be computed (lines 3, 7 and 8 of Algorithm 1).

3.5 Specialisation for rigid transformations

In this section tighter traced volumes will be derived specifically for the group of rigid transformations. Using these tighter traced volumes will result in a faster algorithm for this special subgroup. The focus will lie on the two-dimensional case.

Planar rotations can be represented by angles $r \in \mathbb{R}$, resulting in a representation φ defined by $\varphi r(x) = (\cos(r)\ x_1 - \sin(r)\ x_2, \sin(r)\ x_1 + \cos(r)\ x_2)$, for $x \in \mathbb{R}^2$.

By expressing x in polar coordinates one can determine the minimum and maximum Cartesian coordinates x gets when the angle r is swept along an interval $R \subset \mathbb{R}$. This results in a block $O(R, x)$. For a block $S \in \mathbb{R}^2$ a traced volume $O(R, S)$ can be obtained by computing the axis-parallel block bounding all $O(R, x)$ for all four corners x of S.

The traced-volume operators for rotation and translation can be composed as $V(R, S) = T(R', O(R'', S))$ where $R = R' \times R''$. Since rotations have unit determinant, the function $q(R)$ is simply a constant function with value 1.

4 Results

The metric introduced in Section 2 and the matching algorithm presented in Section 3 work in any dimension. In this section, the results obtained by running a series of tests on pairs of feature patterns obtained from 2D images will be shown. Each test consists of a pair of images depicting similar objects. Some are different views of the same object, some examples consist of an image and a transformed subimage. Implementations of our method for both the one-way Hausdorff distance κ_h and the one-way difference κ_a were tested.

Each example has a natural transformation group associated with it. Examples in which relatively flat three-dimensional objects are seen from different views with a large focal distance ask for matching under affine transformations. Different copies of two-dimensional drawings, on the other hand, ask for rigid transformation. Finally, assuming that three-dimensional objects have the same orientation with respect to the camera, translation or translation in combination with scaling can be used.

4.1 Input data

Figure 1–10 depict the tests. For each figure, the pictures above left and above right, depict the grey-scale images on which feature patterns A and B are superimposed, having white and black dots, respectively. The patterns A and B were extracted using either edge-detection or corner-detection. For each test, the patterns A and B served as inputs.

Figure 1 above right shows a bomber with markings. The markings were cut and cropped manually, resulting in the grey image above left. Figure 2 above right shows a machine with a connector. The neighbouring image shows a different but similar connector. Figures 3 and 4 both depict three-view drawings of a plane. For these two, the images on the left are rotated and translated versions of the top view from the right image. Figure 5 contains images of two planes of the same type, but with different paintings. They are shown from approximately the same side but at slightly different distances. Figure 6 depicts contains pictures of the same plane taken from different side-views. Figure 7 shows two frontal views of the same plane taken from different distances. Figure 8 has two different views of the same fighter. Figure 9 shows a picture of the Mir in varying distance and orientation. Figure 10 shows two different views of the same reconnaissance plane.

In some cases the patterns consists of edges, in others they consist of corners. Figure 1–7 show edge-patterns obtained by Sobel edge-detection followed by thresholding. Figure 8–10 show corners which were indicated manually. The patterns were presented to the Hausdorff matcher as point patterns. For the absolute difference matcher, the point sets were converted to sets of cubes having a radius of 5 pixels for edge-features and 40 pixels for corner-features.

4.2 Parameter settings

The fact that the Hausdorff distance is not invariant under affine transformations, and scaling in particular, cannot be ignored during the tests. Allowing the affine transformations to approach a scaling factor of zero, produces degenerate results for the one-way Hausdorff distance. This could be expected since the scaling of a set A to fit in a small spherical neighbourhood of some

point of B allows the one-way Hausdorff distance to be arbitrarily small. Since such degenerate matchings are not very interesting in a comparison, the method using the one-way Hausdorff distance had its transformation space restricted to exclude arbitrary small scalings. Table 2 shows which initial parameters were used to constrain the linear part of transformation space for both implementations. In each test the accuracy was set to one pixel. This means that the reported matches differ at most the equivalent of one pixel with a global optimum match.

4.3 Matches and statistics

The matches resulting from each test were visualised using both the patterns and the grey-scale images. The middle row of Figure 1–10 show the transformed pattern A superimposed on the pattern B. The bottom row of each figure shows the average grey-scale image of the transformed A grey-scale image and the B grey-scale image. The middle left and bottom left images show the transformation computed by the one-way Hausdorff matching algorithm, while the middle right and bottom right images show the results obtained by the one-way absolute difference.

Table 3 shows, for each test, the transformation group which was used for matching, and the cardinalities of the patterns. The group $\mathbf{U}^2$ mentioned in the table denotes scaling combined with translation, both in two directions. Table 4 contains test-statistics for both the one-way Hausdorff distance and the absolute difference. This table is included only to give an indication of the number of cells and the processing times. Comparing the statistics has limited meaning since, in principle, the implementations solve two different problems. Even though some results obtained using the one-way function difference took longer to compute, the results were accurate, while the one-way Hausdorff distance produced "false" matches in these cases. Moreover, for the examples involving scaling (tests five to ten), the transformation space for the one-way Hausdorff distance is only half the size of that for the absolute difference, because of the restriction of the scaling.

4.4 Implementation

Both the one-way Hausdorff version of our technique, described in [10] and the one-way absolute function difference variant described in the current paper were implemented in C++. The implementations use as many of the same libraries and basic data-structures as possible. Both programs were compiled using the the SGI Delta/C++ compiler. The tests were executed on a Silicon Graphics Indy workstation having a MIPS R5000 processor and 64 MB of memory.

method	rotation	scaling	linear
one-way Hausdorff distance	$[0, 2\pi]$	$[\frac{1}{2}, 1.0]^2$	$-1 \leq L_{ij} \leq 1, \frac{1}{4} \leq \det(L) \leq 1$
one-way absolute difference	$[0, 2\pi]$	$[0, 1.0]^2$	$-1 \leq L_{ij} \leq 1, 0 \leq \det(L) \leq 1$

Table 2: Initial parameters for linear transformation components.

test figure	group	# features A	# features B
1	$\mathbf{T}^2$	60	1084
2	$\mathbf{T}^2$	110	773
3	$\mathbf{SM}^2$	189	439
4	$\mathbf{SM}^2$	203	637
5	$\mathbf{U}^2$	175	122
6	$\mathbf{U}^2$	308	317
7	$\mathbf{U}^2$	717	740
8	$\mathbf{A}^2$	16	17
9	$\mathbf{A}^2$	23	29
10	$\mathbf{A}^2$	20	20

Table 3: Overview of the tests.

4.5 Discussion

Table 5 shows the determinants of the linear parts of the transformations computed by both methods, for all tests involving scaling. The table shows that the Hausdorff distance is inclined to select the transformation with a relatively low determinant for the linear part, while the absolute difference works well despite the fact that the determinant was allowed to reach zero. Thus the one-way Hausdorff distance can be used only for groups containing scaling when severe restrictions can be put on transformation space a priori. Things can be made better by using the (two-way) Hausdorff distance. However, this distance does not work with outliers and missing parts. To remedy this, the partial Hausdorff distance could be used. However, this requires knowledge of the percentage of point which are expected to lie in each other's neighbourhoods. For accurate matching, the absolute function difference must be preferred over the partial Hausdorff distance.

5 Conclusion

Our pattern matching algorithm is not specialised or optimised for any specific application whatsoever and does not make any assumptions about optimal transformations like matching on the centroid or other reference points. The processing times given in Section 4 were obtained using a non-optimised

test figure	# cells		# seconds	
	Hausdorff	absolute	Hausdorff	absolute
1	573	791	53.8	7.2
2	409	709	57.1	7.7
3	641	7275	127.4	233.1
4	961	29137	250.2	1020.0
5	3253	74487	294.9	867.2
6	975	76917	237.5	1884.4
7	641	8413	568.2	533.5
8	543679	23057	2449.7	24.6
9	375673	177997	2928.0	290.4
10	1167449	70321	7183.8	101.2

Table 4: Statistics for both tests.

test figure	determinant	
	Hausdorff	absolute
5	0.35	0.42
6	0.34	0.56
7	0.44	0.61
8	0.34	0.63
9	0.44	0.55
10	0.43	0.50

Table 5: Determinants of the linear components of reported matches.

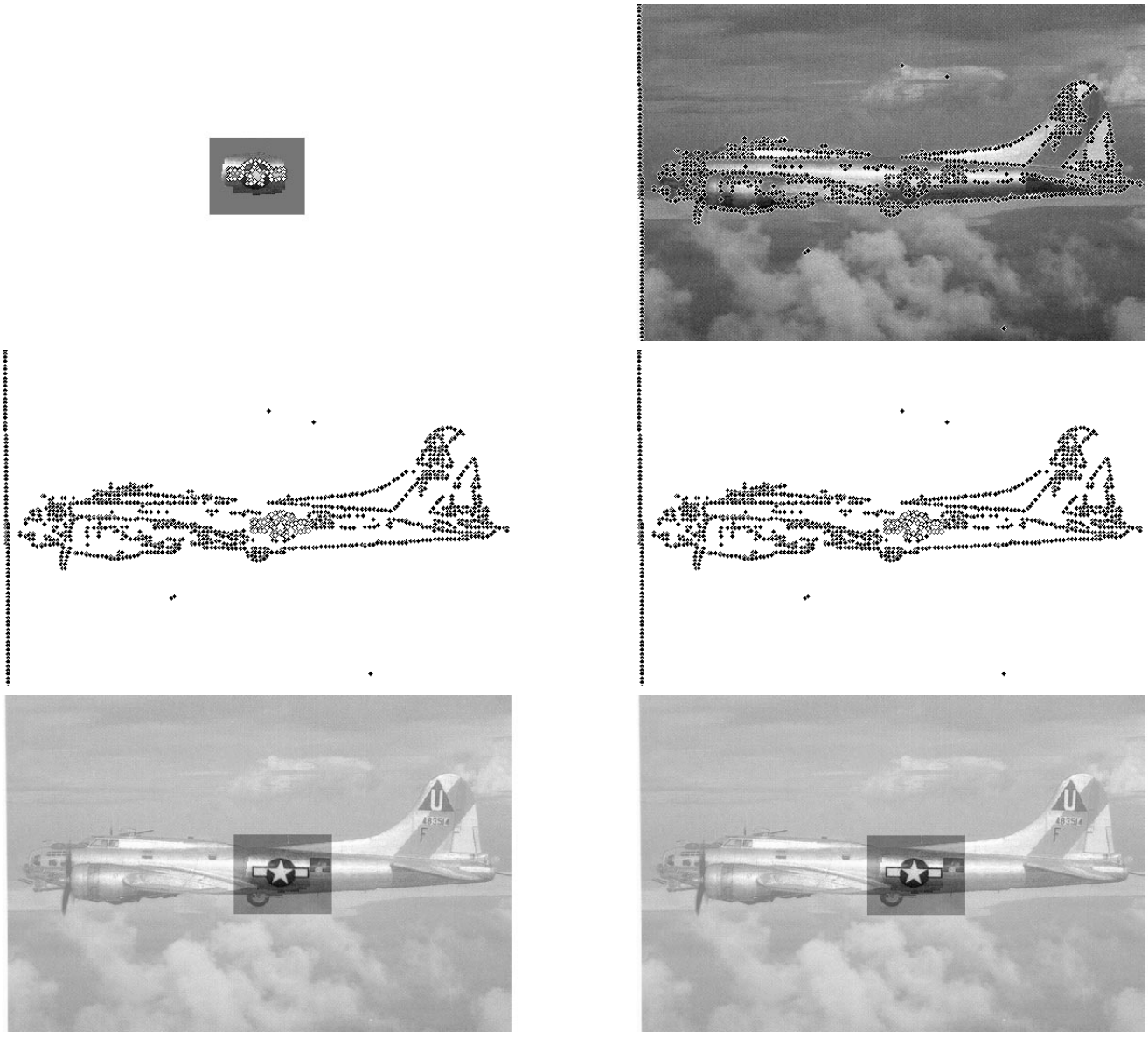

Figure 1: B17 bomber with subimage.

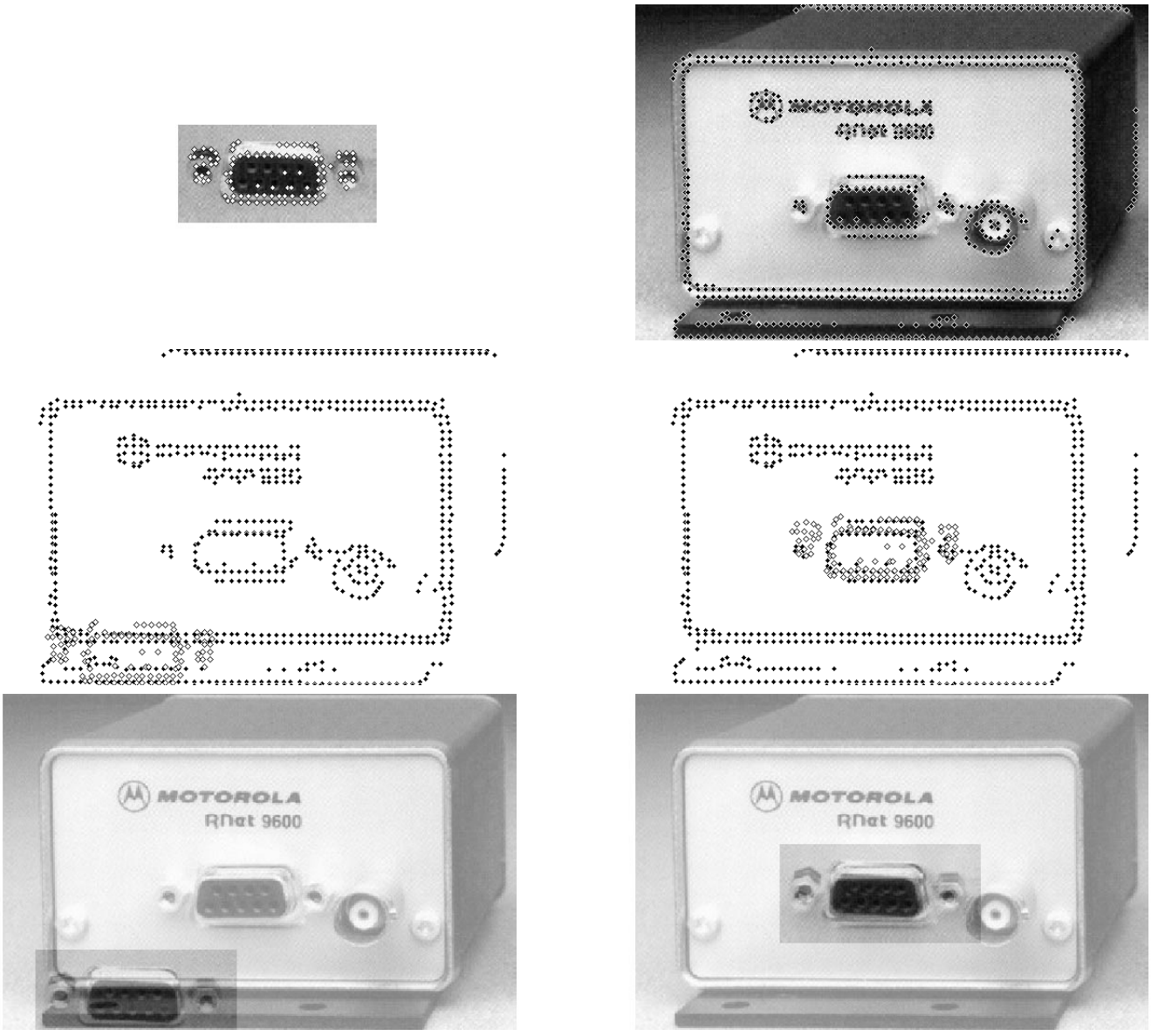

Figure 2: Device with a different connector.

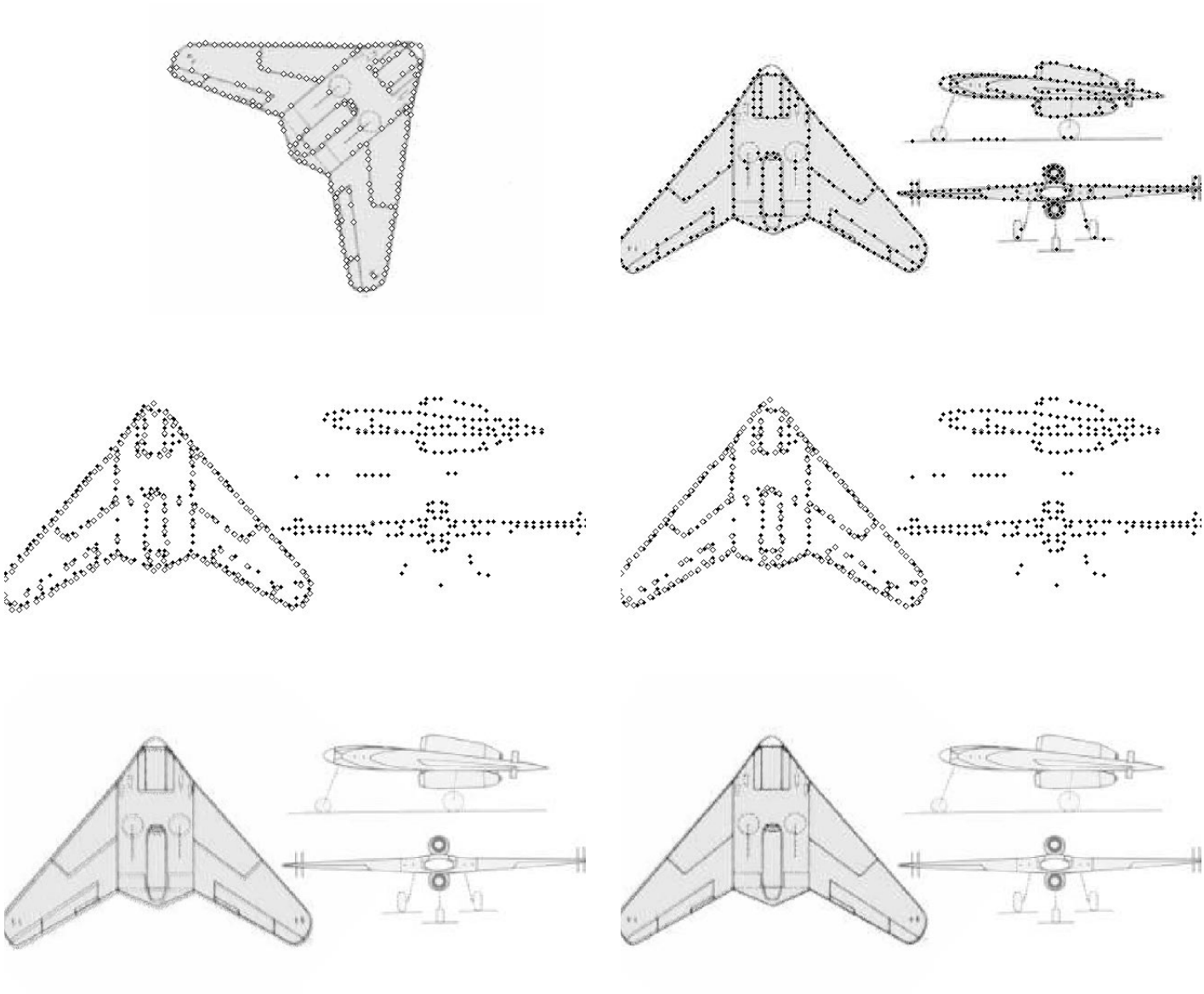

Figure 3: Three-view drawing with subimage.

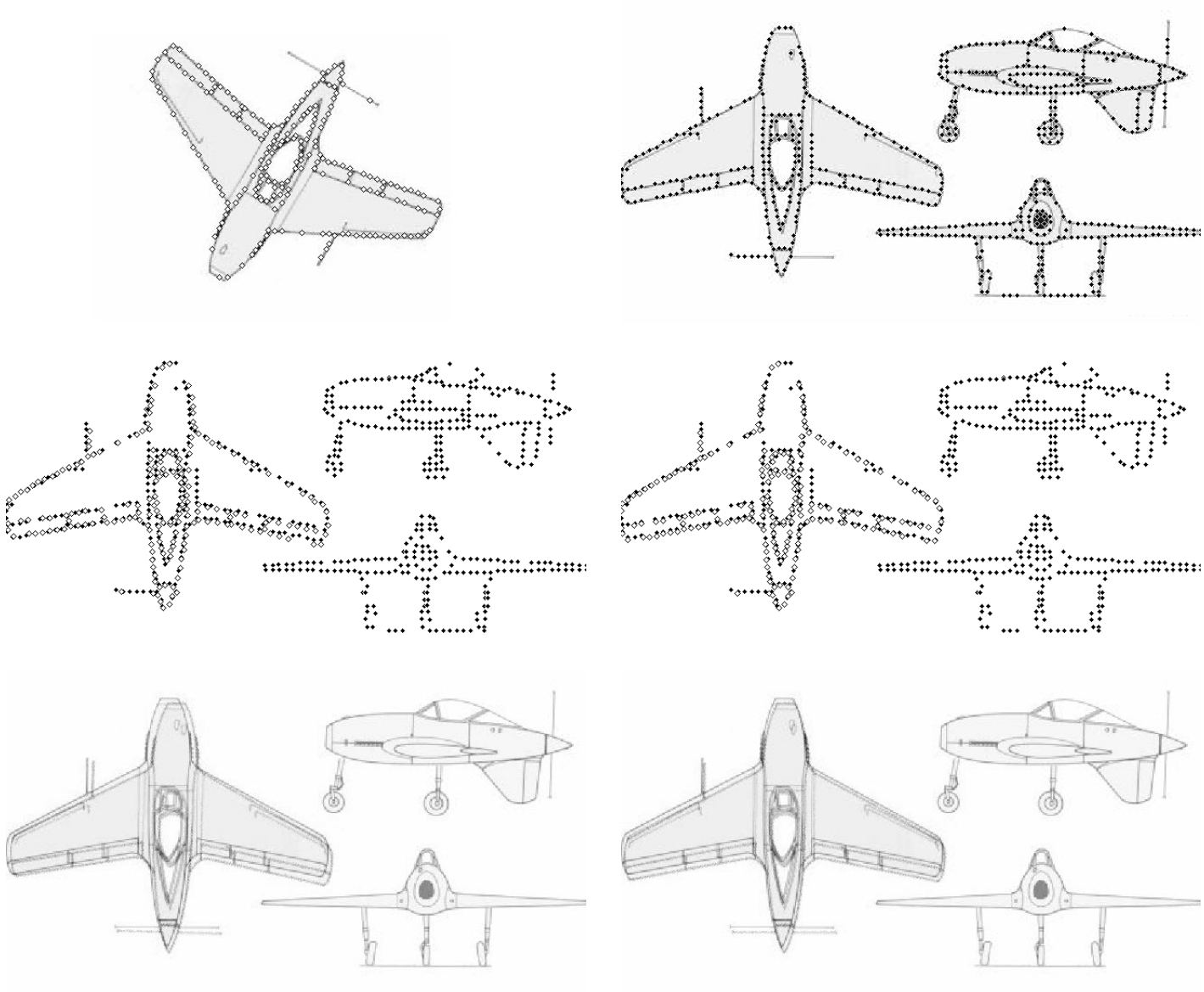

Figure 4: Three-view drawing with subimage.

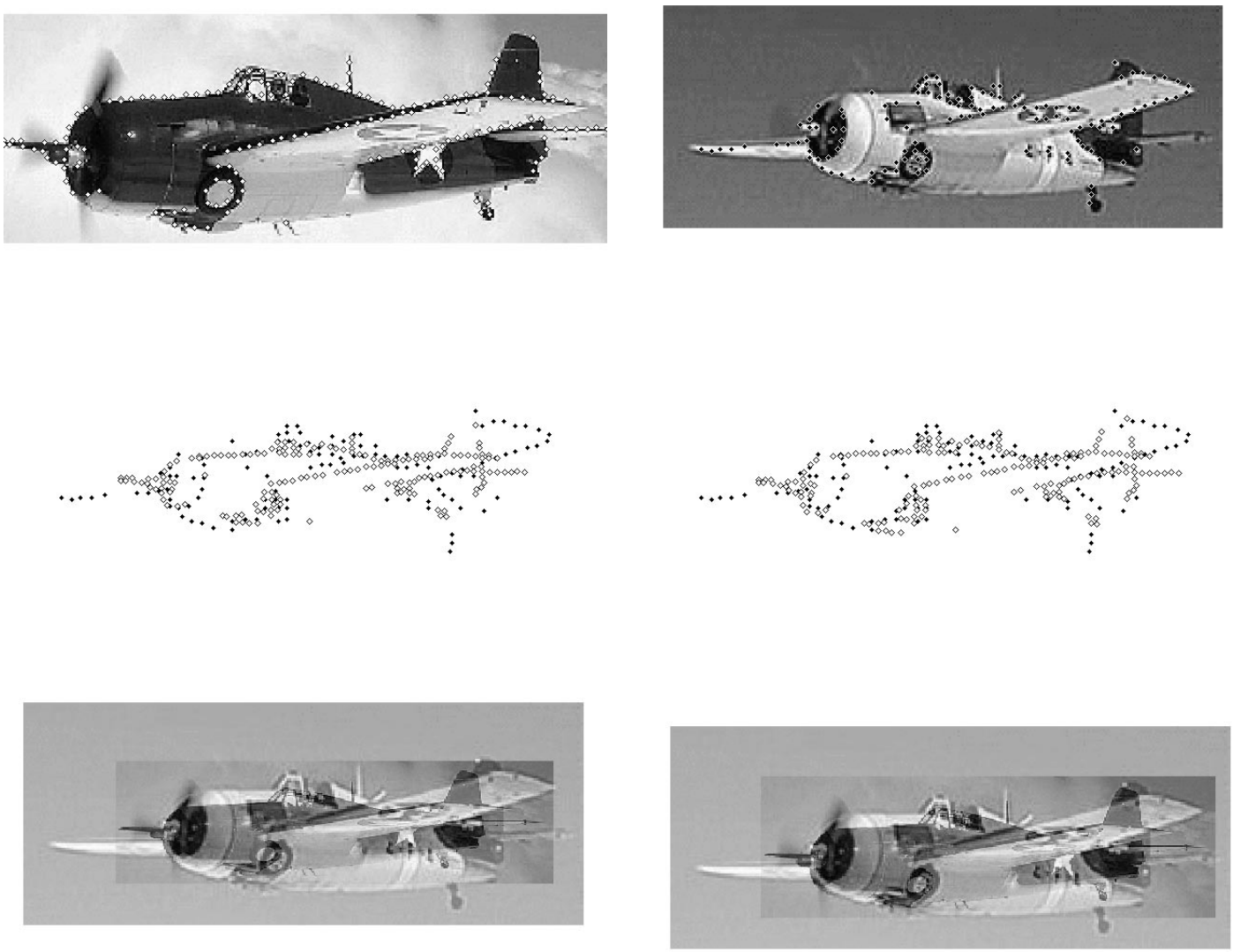

Figure 5: Two different planes of the same type.

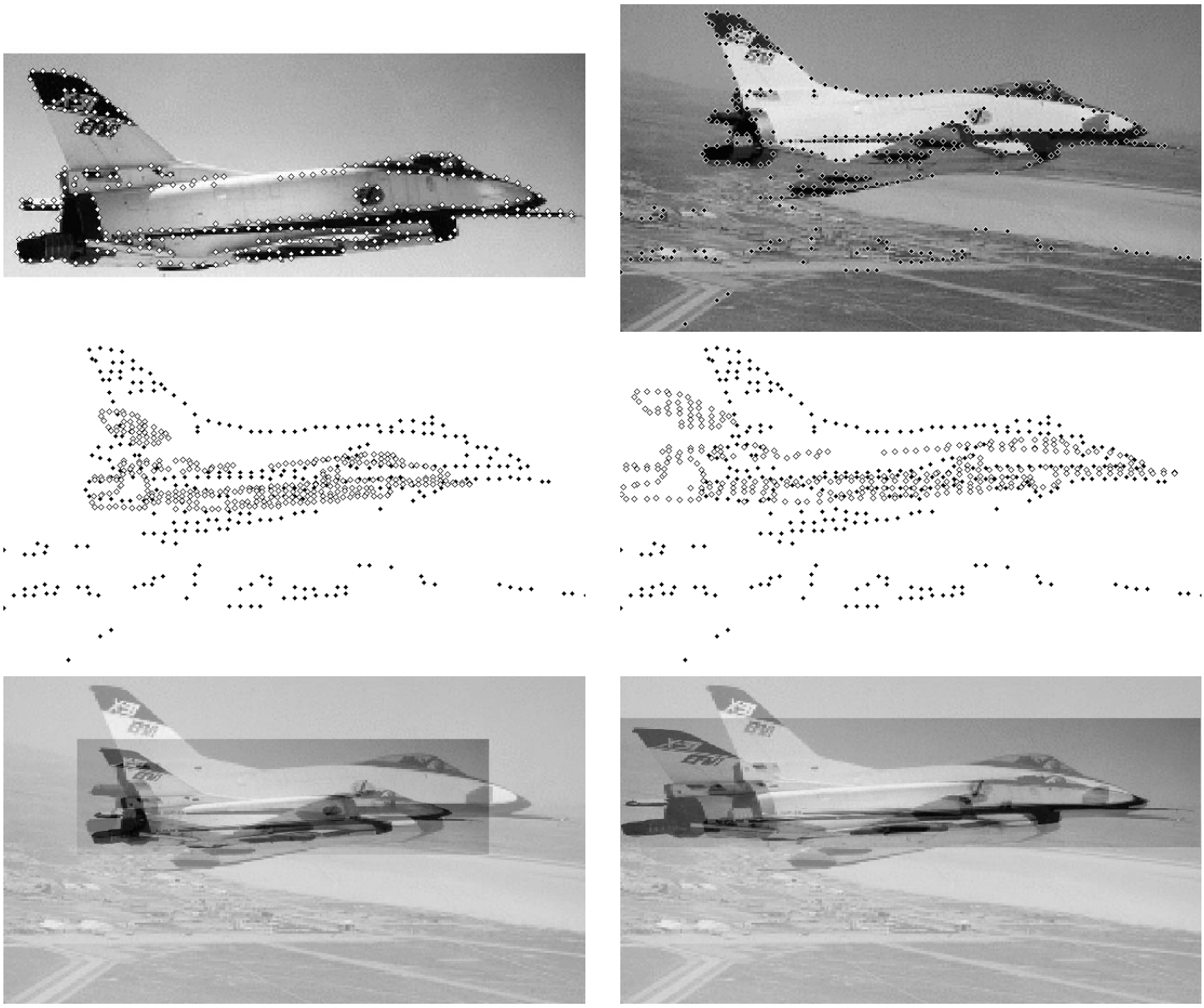

Figure 6: Different side views of the same plane.

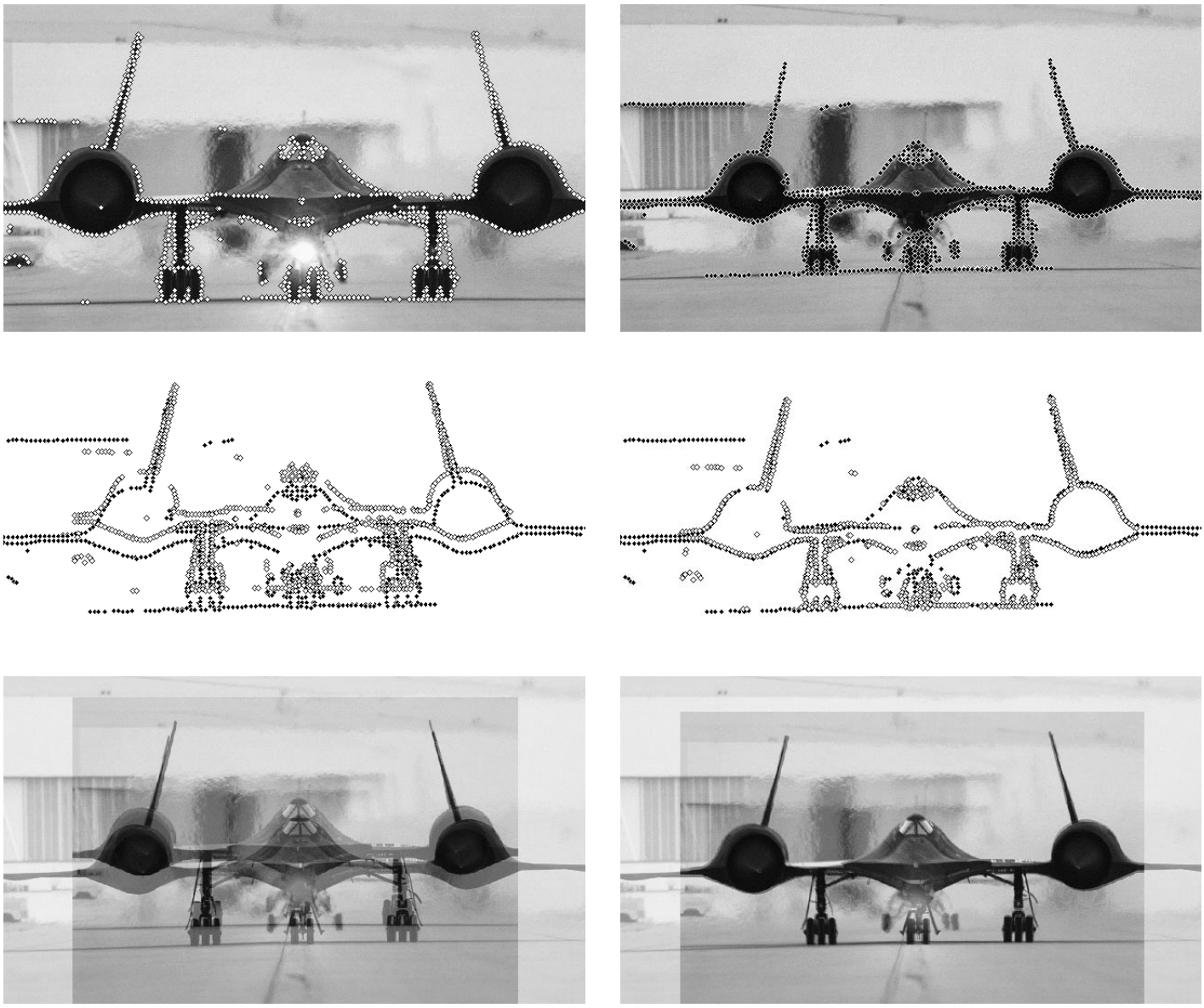

Figure 7: Two different front views of the same plane.

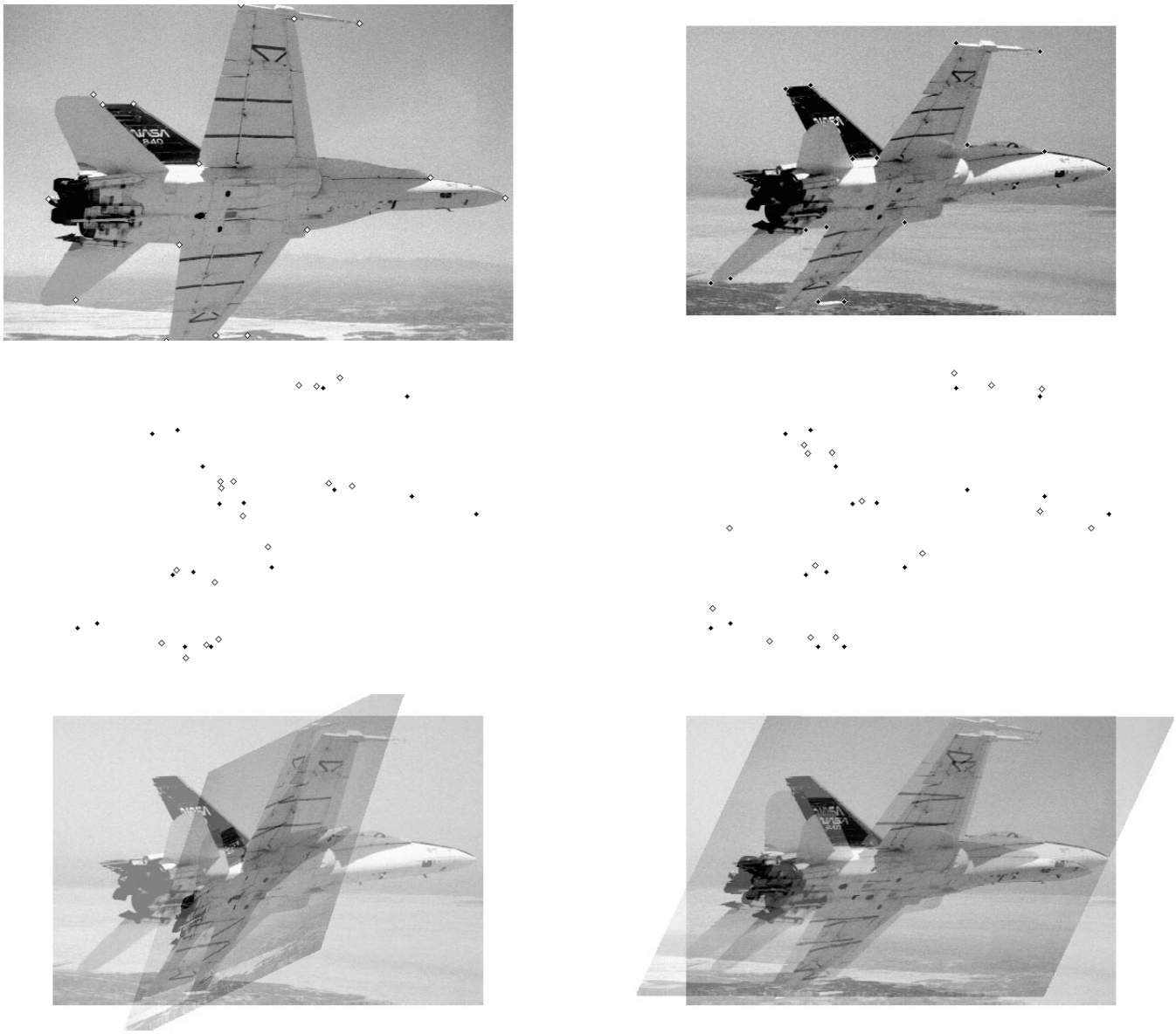

Figure 8: Two different view of the same F18 fighter.

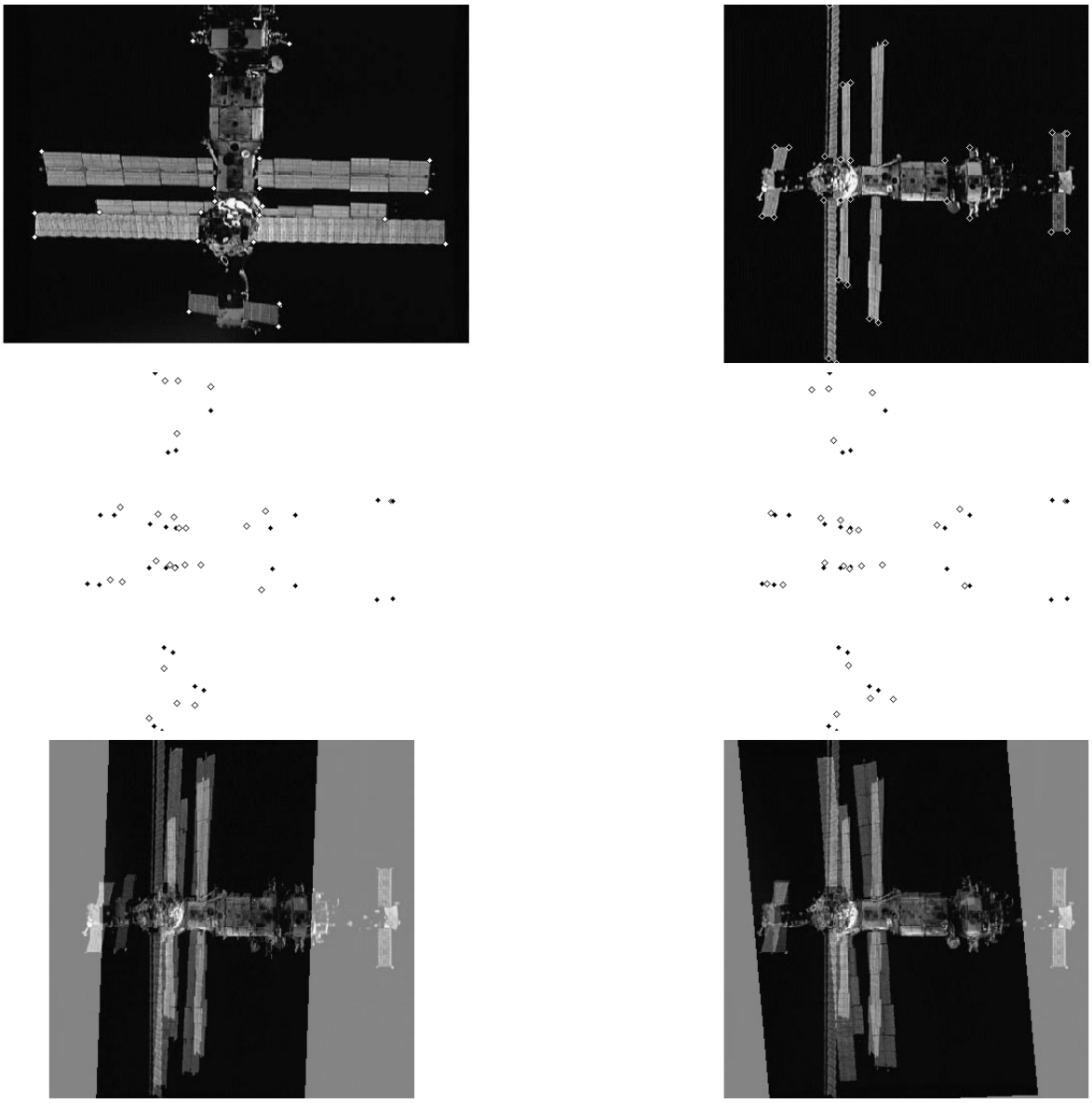

Figure 9: Two different views of the Mir station.

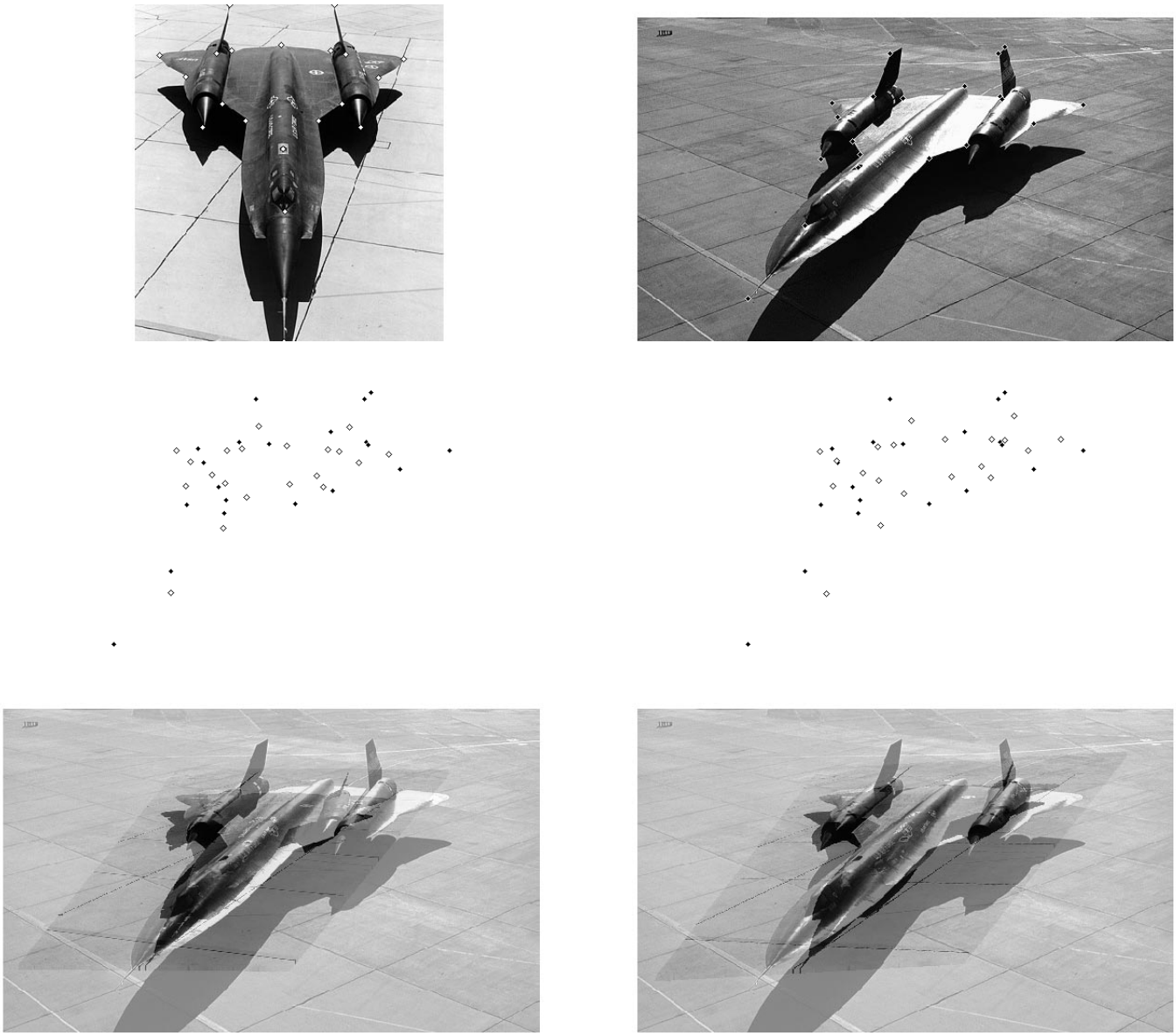

Figure 10: Two different views of the same reconnaissance plane.

algorithm which uses floating-point arithmetic and arrays of floats for representing vectors. Also, minimal restrictions on transformation space were made beforehand. In contrast with many application-dependent heuristics, our matching method uses a mathematically well-founded metric for comparing patterns. In addition, this pattern-metric behaves well with respect to perturbation, outlying and absent parts and is invariant under affine transformation. Our method also works for a very broad class of patterns, namely the class of solid functions.

It must be emphasised that our method is not a heuristic: it guarantees that the computed transformation matches the patterns to a distance not further than any fixed constant from the actual minimum value of the pattern-metric. Restriction of transformation space, as a result of assumptions that can be made in a specific application domain, can be incorporated in the algorithm in an easy way, resulting in a fast optimised algorithm for that purpose.

In Section 2, a new pattern-similarity measure called the absolute difference, was introduced. Like the well-known Hausdorff distance, this measure satisfies the metric properties and is robust against small pattern-deformations. Unlike the Hausdorff distance, the absolute difference is robust against outlying and absent parts, despite the fact that it satisfies the metric properties. An important advantage of the absolute difference over the partial Hausdorff distance is that it does not depend on a parameter. In contrast with the symmetric difference, the absolute difference is affine-invariant, making it suitable for pattern matching under affine transformations, without the need of restricting the search space.

In Section 3, the general pattern matching technique was introduced. This method avoids having to process all local minima by subdividing transformation space hierarchically. The method is very general. It can be applied to the symmetric difference, the Hausdorff distance, the absolute difference, and other metrics. In addition, it can be used on patterns in vector spaces of any dimension, and various classes of transformations.

Section 4 described experiments using an implementation of our method for both the Hausdorff distance and the absolute difference. The tests showed that the absolute difference matches more reliable than the Hausdorff distance for transformation groups involving scaling. Considering the high complexity of the minimisation problem (see Section 1), the execution times listed in Section 4 seem reasonable.

A Definitions

A.1 Groups

A group is a set G together with a composition (denoted by juxtaposition) such that:

(i) $(gh)k = g(hk)$ for all $g, h, k \in G$.

(ii) There exists an element $e \in G$ for which $ge = g = eg$.

(iii) Each element $g \in G$ has an inverse g^{-1} for which $g^{-1}g = e = gg^{-1}$.

The element e is unique and is called the identity of G. The mapping $a \mapsto a^{-1}$ is called the inversion mapping.

A.2 Topological Groups

A topological group is a topological space having a group structure with continuous composition and inversion mappings. A topological transformation group is a topological group together with a space X such that:

(i) $g(hx) = (gh)x$ for all $g, h \in G$ and $x \in X$.

(ii) $ex = x$ for all $x \in X$.

The mapping $(g, x) \mapsto gx$ is called the action of G on X. For more details on topological transformation groups see Bredon [5].

A.3 Metric spaces

A metric space (X, d) is a set X with a function d that satisfies the following properties:

(i) $d(x, y) \geq 0$.

(ii) $d(x, y) = 0$ if and only if $x = y$.

(iii) $d(x, y) = d(y, x)$.

(iv) $d(x, z) \leq d(x, y) + d(y, z)$.

The function d is called a metric. Property (iv) is called the triangle inequality.

References

[1] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. *Proceedings of the 12th Symposium on Computational Geometry*, pages 301–310, 1996.

[2] Alon Efrat and Matthew J. Katz. Computing fair and bottleneck matchings in geometric graphs. Technical report, May 1996.

[3] M. D. Atkinson. An optimal algorithm for geometrical congruence. *J. Algorithms*, 8:159–172, 1987.

[4] N. Ayache and B. Faverjon. Efficient registration of stereo by matching graph descriptors of edge segments. *International Journal of Computer Vision*, pages 107–131, 1987.

[5] Glen E. Bredon. *Introduction to Compact Transformation Groups*, volume 46 of *Pure and Applied Mathematics*. Academic Press, 1972.

[6] E. Cuchet, J. Knoplioch, D. Dormont, and C. Marsault. Registration in neurosurgery and neuroradiotherapy applications. In *Proceedings of the 2nd International Symposium on Medical Robotics and Computer Assisted Surgery, Baltimore, Maryland*, 1995.

[7] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the hausdorff distance. Technical Report 91-1211, Cornell University, 1991.

[8] Daniel P. Huttenlocher, Klara Kedem, and Micha Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9:267–291, 1993.

[9] Daniel P. Huttenlocher and William J. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. Technical Report 92-1321, Cornell University, 1992.

[10] Michiel Hagedoorn and Remco C. Veltkamp. A general method for partial point set matching. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 406–408, 1997.

[11] Helmut Alt, Bernd Behrends, and Johannes Blomer. Approximate matching of polygonal shapes. *Annals of Mathematics and Artificial Intelligence*, pages 251–265, 1995.

[12] Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.

[13] Helmut Alt and Michael Godeau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, pages 75–91, 1995.

[14] Helmut Alt, Ulrich Fuchs, Gunter Rote, and Gerald Weber. Matching convex shapes with respect to the symmetric difference. Technical Report B 92-03, Freie Universität Berlin, April 1996.

[15] Jin-Long Chen and George C. Stockman. Determining pose of 3d objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1), January 1996.

[16] L. Paul Chew, Michael T. Goodrich, Daniel P. Huttenlocher, Klara Kedem, Jon M. Kleinberg, and Dina Kravets. Geometric pattern matching under Euclidean motion. In *Fifth Canadian Conference on Computational Geometry*, pages 151–156, 1993.

[17] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.

[18] Rajiv Mehrotra and James Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, pages 55–62, 1995.

[19] O. Aichholzer, H. Alt, and G. Rote. Matching shapes with a reference point. In *International Journal of Computational Geometry and Applications*, volume 7, pages 349–363, August 1997.

[20] P. J. de Rezende and D. T. Lee. Point set pattern matching in d-dimensions. *Algorithmica*, 13:387–404, 1995.

[21] William Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. Lecture Notes in Computer Science. Springer-Verlag, 1996.

[22] William J. Rucklidge. Lower bounds for the complexity of the Hausdorff distance. In *Proceedings of the Fifth Canadian Conference on Computational Geometry*, pages 145–150, 1993.

[23] Kasturi R. Varadarajan. Approximating monotone polygonal curves using the uniform metric. In *Proceedings of the 12th Annual Symposium on Computational Geometry*, pages 311–318, 1996.

[24] Warren F. Gardner and Daryl T. Lawton. Interactive model-based vehicle tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11), November 1996.

[25] J. West, J. M. Fitzpatrick, M. Wang, B. Dawant, Jr. C. Maurer, R. Kessler, R Maciunas, C. Barillot, D. Lemoine, A. Collignon,

F. Maes, P. Suetens, D. Vandermeulen, P. van den Elsen, S. Napel, T. Sumanaweera, B. Hardkness, P. Hemler, D. Hill, D. Hawkes, C. Studholme, J. A. Maintz, M. Viergever, G. Malandain, X. Pennec, M. Noz, Jr. G. Maguire, M. Pollack, C. Pelizzari, R. Robb, D. Hanson, and R. Woods. Comparison and evaluation of retrospective intermodality brain image registration techniques. *Journal of Computer Assisted Tomography*, 21(4):554–566, 1997.

[26] Haim J. Wolfson. Model-based object recognition by geometric hashing. *First European Conference on Computer Vision*, 427:526–536, 1990.

[27] Yanhong Li, Daniel Lopresti, George Nagy, and Andrew Tomkins. Validation of image defect models for optical character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2), February 1996.