Reliable and efficient pattern matching using an affine invariant metric

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

Abstract

We present a new pattern similarity measure that behaves well under affine transformations. Our similarity measure is useful for pattern matching since it is defined on patterns with multiple components, satisfies the metric properties, is invariant under affine transformations, and is robust with respect to perturbation and occlusion. We give an algorithm, based on hierarchical subdivision of transformation space, which minimises our measure under the group of affine transformations, given two patterns. In addition, we present results obtained using an implementation of this algorithm.

1 Introduction

In applications such as pose determination [15], object recognition [22], vehicle tracking [21], optical character recognition [19], stereo matching [5], and content based image retrieval [17], a major problem is finding a geometric transformation that matches part of a pattern to some part of another pattern. In practice, the patterns typically consist of features extracted from technical drawings, sketches, handwriting, and digitised images.

In this paper, we take an abstract approach to geometric pattern matching. In our model, patterns are subsets of a finite-dimensional Euclidean space and pattern matching is the problem of finding geometric transformations that minimise a similarity measure between a transformed pattern and another pattern.

For pattern matching tasks like object recognition and pose determination we want to use a similarity measure that satisfies a number of properties. First, we want the similarity measure to be a metric. Second, the similarity measure must be invariant for the group of geometrical transformations under consideration. Third, the measure should be robust for perturbation. This means that the less we deform a pattern, the closer the deformed pattern becomes to the original pattern, under the similarity measure. Fourth, the similarity measure must behave well in case of occlusion. This implies that covering a pattern with a

small region or cutting a small region from a pattern should result in only a small difference of the similarity measure. Besides a reliable similarity measure, an efficient minimisation algorithm is desired.

Heuristics used in most practical methods make domain dependent assumptions. A well known heuristic in pattern matching is aligning the centroids of two patterns. This eliminates translation, allowing another method to solve for the remaining degrees of freedom. In applications where patterns suffer from a significant amount of occlusion, 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, or local search techniques to find a matching transformation. However, symmetric or almost symmetric patterns falsify the convexity assumption, for any reasonable similarity measure. Since we want to perform pattern matching reliably, we refrain from using these error prone heuristics.

Let us review previous applications of computational geometry to pattern matching. *Exact* pattern matching algorithms decide whether or not transformations exist that map a pattern precisely onto another pattern. To put this in our framework, exact pattern matching uses the metric which has value zero if two patterns are equal and value one otherwise. Clearly, this metric can be used for any class of patterns and is invariant under every transformation group. Unfortunately, it is not at all robust for perturbation and occlusion. Even if two patterns differ at only one point, the metric has value one. In most practical applications, measurement errors and limited numerical precision render exact matching algorithms unreliable. Efficient algorithms are known for the exact matching of two point sets each having n elements. Alt et al. [11] show that under planar motions, matches can be reported in $O(n \log n)$ time. Sprinzak and Werman [14] show how to reduce exact affine matching to matching under motions in linear time, resulting in an $O(n \log n)$ time algorithm.

Bottleneck matching achieves increased robustness in case of perturbations and roundoff errors by considering point set correspondences. It is based on the minimisation of a similarity measure which we will call the *bottleneck distance*. Given two point sets, this distance minimises, over all bijections between the point sets, the maximum distance between any point in one set and its image in the other set. Variants on bottleneck matching include minimum weight matching, uniform matching, and minimum deviation matching. The bottleneck distance and its variants are by definition invariant under Euclidean isometries. They are also robust for perturbation: small movements of points have little impact on the quality of a correspondence. What makes the bottleneck distance unsuitable for our purposes is that, in principle, only equal-sized finite point sets can be matched. This means the method cannot handle addition or removal of points. For planar point sets of size n, the computation of the bottleneck distance can be performed in $O(n^{1.5} \log n)$ time, while the bottleneck distance can be minimised under translation in $O(n^5 \log^2 n)$ time, see Efrat et al. [1, 2].

Given two subsets of the plane whose boundaries are curves, the *Fréchet distance* is defined as the "maximum distance" over all parametrisations of the boundaries. By definition, the Fréchet distance is invariant under Euclidean

isometries. In addition, it is robust for perturbations. However, it is unsuitable for pattern matching with occlusion. The Fréchet distance of two polygonal curves with m and n defining points, can be computed in $O(mn \log mn)$ time; see Alt and Godau [12].

Much work has been done on matching patterns using the *Hausdorff distance*. For patterns being compact sets, the Hausdorff distance equals the maximum over all points in both patterns of the distance to the closest point in the other pattern. The Hausdorff distance is a metric. In contrast with correspondence based matching, the Hausdorff distance does not always produce bijections between point sets. Multiple points in one pattern may be associated with a single point in another pattern. The Hausdorff distance is invariant under isometries, but not under affine transformations. Although robust for perturbation, it is very sensitive to occlusion: adding or deleting points can result a significant change of the Hausdorff distance. Various algorithms are known for finite point sets of sizes m and n. The Hausdorff distance itself can be computed in $O((m+n)\log(m+n))$ time; see Alt et al. [3]. Huttenlocher et al. [7] derive an $O((mn)^2((m+n)^{1+\epsilon})$ time algorithm for minimisation under translation in the plane. For planar motion, an $O((m+n)^6 \log mn)$ time algorithm was found by Huttenlocher et al. [13].

The *symmetric difference* equals the volume of the symmetric set difference. Matching using the symmetric difference under volume preserving affine transformations is equivalent to maximising the volume of the intersection, or the area of overlap in the planar case. Mount [8] investigates properties of the area of overlap as a function of translation. Alt [4] gives efficient algorithms for finding pseudo optimal matches for the area of overlap under various transformations including translation and Euclidean motion. An $O((m+n)\log(m+n))$ time algorithm for finding a translation maximising the area of overlap between convex polygons determined by m and n points was found by de Berg et al. [9]. The symmetric difference is robust for perturbation as well as occlusion. However, the symmetric difference is not invariant under full affine transformations. It is only invariant under volume preserving affine transformations, which we will call "special affine transformations".

In this paper, a variation on the symmetric difference, called the *absolute difference*, is proposed as a similarity measure. Like the symmetric difference, this similarity measure is a metric robust for perturbation and occlusion. In addition, the absolute difference is invariant under full affine transformations, making it a particularly well behaved similarity measure under transformations with a scaling component.

The pattern matching algorithm 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 when the minima are concentrated in a limited number of small clusters in transformation space. Under such circumstances our algorithm is likely to eliminate large subsets of transformation space at an early stage. In [10], we presented an hierarchical transformation space subdivision algorithm for matching using the partial Hausdorff distance. Here, we present a pattern matching algorithm based on the absolute difference.

The algorithm can be used to approximate the minimum absolute difference with any desired accuracy, reporting a near minimal transformation in the process. The algorithm is general; it works for any dimension, and various classes of transformations, including translations, scalings combined with translations, motions, similarity transformations, volume preserving affine transformations, and affine transformations in general.

2 Pattern similarity measures

In this paper, patterns are subsets of d-dimensional Euclidean space R^d. Given a group of geometric transformations, the *shape* of a pattern is the pattern modulo the action of the transformation group. Section 2.1 discusses transformation groups that are fundamental in geometric pattern matching. Minimisation of pattern metrics under the action of a transformation group is discussed in Section 2.2. Section 2.3 discusses a number of desirable properties for similarity measures. Section 2.4 gives various pattern similarity measures, including the absolute difference, the new similarity measure we introduce in this paper. Finally, Section 2.5 investigates the absolute difference in more detail.

2.1 Transformation groups

Many practical object recognition and pose determination tasks can be solved by minimisation of a similarity measure under a group of transformations. The transformation group depends on the problem. In optical character recognition one might consider the class of translations. If we want to recognise objects in images obtained by aerial photography, matching under two-dimensional motions can be useful. Affine transformations are often used as an approximation of camera viewpoint changes under weak perspective.

The full linear group $\mathrm{GL}(d)$ consists of all linear functions in R^d having nonzero determinant. The special linear group $\mathrm{SL}(d)$ is a subgroup of the full linear group. It consists of all elements of the full linear group having determinant 1. The full orthogonal group $\mathrm{O}(d)$ consists of all linear transformations that preserve the Euclidean norm. That is $\|g(x)\| = \|x\|$ for all $g \in \mathrm{O}(d)$ and all $x \in \mathrm{R}^d$. The special orthogonal group $\mathrm{SO}(d)$ consists of those linear functions both member of $\mathrm{O}(d)$ and $\mathrm{SL}(d)$. Another linear subgroup, the group of scalings $\mathrm{S}(d)$, consists of functions that multiply each coordinate of a point with a nonzero constant. The group of uniform scalings $\mathrm{U}(d)$ is the group of scalings having the same scaling factor for each coordinate.

The composition $H \circ G$ of two sets of functions G and H is the collection of all compositions hg, for $g \in G$ and $h \in H$. New groups can be formed from the former ones by composition with the translations $\mathrm{T}(d)$. This way, the full affine transformations are defined as $\mathrm{Aff}(d) = \mathrm{T}(d) \circ \mathrm{GL}(d)$. These transformations preserve parallelism of hyperplanes. The subgroup of special affine transformations $\mathrm{SAff}(d)$ is the composition of the special linear group $\mathrm{SL}(d)$ with the translations $\mathrm{T}(d)$. These transformations preserve volume. The isometries, a

subgroup of $\mathrm{Aff}(d)$, can be written as the composition $\mathrm{Iso}(d) = \mathrm{T}(d) \circ \mathrm{O}(d)$. This group consists of all distance preserving mappings. The group of motions $\mathrm{Mot}(d)$ can be constructed as all special orthogonal transformations $\mathrm{SO}(d)$ composed with translations $\mathrm{T}(d)$. The group of similarity transformations $\mathrm{Sim}(d)$ is the collection of motions $\mathrm{Mot}(d)$ composed with uniform scalings with positive scaling coefficient. Finally, $\mathrm{TS}(d)$ denotes scalings combined with translations.

2.2 Transforming patterns

We will now discuss transformation groups acting on patterns. If G is a transformation group for R^d, define the action of $g \in G$ on a *subset* $A \subseteq \mathrm{R}^d$ by

$$g[A] = \{\, g(a) \mid a \in A \,\}.$$

This makes G a transformation group for the power set of R^d. Suppose that $\mathcal{P}$ is a collection of patterns closed under the action of G, i.e. $A \in \mathcal{P}$ implies $g[A] \in \mathcal{P}$ for all $g \in G$. Then G is a transformation group for the collection of patterns $\mathcal{P}$, under the previously defined action.

A collection of patterns $\mathcal{P}$ and a transformation group G determine a family of shapes. For a pattern $A \in \mathcal{P}$, the corresponding shape is the orbit

$$G[A] = \{\, g[A] \mid g \in G \,\}.$$

This is similar to the definition of shape adopted by Small in [20], where two patterns A and B are said to have the same shape if there exists a similarity transformation $g \in \mathrm{Sim}(d)$ such that $g[A] = B$. Our notion of shape is more general in the sense that it is defined with respect to any chosen transformation group G. The group G defines an equivalence relation on $\mathcal{P}$, where A and B are equivalent if there exists a transformation $g \in G$ such that $g[A] = B$. Therefore, the shape $G[A]$ of a pattern $A \in \mathcal{P}$ is in fact an equivalence class in $\mathcal{P}$. Now the family of shapes $\mathcal{P}/G$ determined by $\mathcal{P}$ and G is the partition

$$\mathcal{P}/G = \{\, G[A] \mid A \in \mathcal{P} \,\}.$$

Let d be a metric on the collection of patterns $\mathcal{P}$. We are interested in the value $d(g[A], B)$ as a function of transformations $g \in G$. Let m be the greatest lowest bound for this set of non-negative real numbers. It is not known whether a transformation $h \in G$ exists such that $d(h[A], B) = m$. The following lemma gives a condition that guarantees the existence of such transformations.

Lemma 1 *Let (M, d) be a metric space; let G be a topological transformation group for M. Suppose for each open ball $B(z, \epsilon) \subseteq M$, there exists a compact subset $H \subseteq G$ such that $B(z, \epsilon) \cap G[z] \subseteq H[z]$. Then for all $x, y \in M$, there exists a $k \in G$ such that $d(k[x], y) \leq d(g[x], y)$ for all $g \in G$.*

Proof. Let x and y be points of M and let $\epsilon = d(x, y)$. Choose a compact subset $H \subseteq G$ such that $B(x, 2\epsilon) \cap G[x] \subseteq H[x]$. Define a function $f : G \to \mathrm{R}$ by $f(g) = d(g[x], y)$.

First, we claim $f(h) \geq r$ for all $h \in H$, implies $f(g) \geq r$ for all $g \in G$. We assume $f(g) < r$ for some $g \in G - H$, and derive a contradiction. Now $f(g) < f(h)$ for all $h \in H$. The set of transformations H contains an element h_0 for which $h_0[x] = x$, giving $d(g[x], y) < d(h_0[x], y) = d(x, y)$. Then, by the triangle inequality $g[x] \in B(y, \epsilon) \subseteq B(x, 2\epsilon)$. Therefore $g[x]$ must lie in $H[x]$, meaning there exists an element $h \in H$ such that $g[x] = h[x]$, giving $f(h) < r$. Contradiction.

The mapping $g \mapsto g[x]$ from G to M is continuous. Using the fact that d is a metric it can be shown that $d : M \times M \to \mathrm{R}$ is a continuous function. Hence, we can write f as a composition of continuous functions. Since H is compact, there exists a $k \in H$ such that $f(k) \leq f(h)$ for all $h \in H$. By our previous claim this implies $f(k) \leq f(g)$ for all $g \in G$, finishing the proof. $\square$

A metric d on a set M is said to be *invariant* for the transformation group G if $d(g[x], g[y]) = d(x, y)$ for all $g \in G$ and $x, y \in M$. The following theorem says that minimisation of an invariant metric on patterns gives a metric on shapes.

Theorem 1 *Let G be a transformation group for a set M; let d be a metric on M invariant for G. Suppose for each pair of points $x, y \in M$, there exists a group element $g \in G$ minimising the value $d(g[x], y)$. Then a metric $\tilde{d}$ on the orbit space M/G is defined by*

$$\tilde{d}(G[x], G[y]) = \min\{\, d(g[x], y) \mid g \in G \,\}.$$

Proof. First, we need to show that $\tilde{d}$ is well defined. An equivalent definition of $\tilde{d}$ is given by

$$\tilde{d}(G[x], G[y]) = \min\{\, d(g[x], h[y]) \mid g, h \in G \,\}.$$

From this definition, it immediately follows that the value of $\tilde{d}$ is independent of the choice of representatives.

Now, we prove the metric properties for $\tilde{d}$. The non-negativity of $\tilde{d}$ follows by definition. Next, we show definiteness $\tilde{d}(G[x], G[y]) = 0 \Leftrightarrow G[x] = G[y]$. The implication from right to left follows by $\tilde{d}(G[x], G[x]) = d(e[x], x) = 0$. Now, suppose $\tilde{d}(G[x], G[y]) = 0$. By assumption, there exists a $h \in G$ for which $d(h[x], y) = 0$. By definiteness of d, we get $h[x] = y$, implying $G[x] = G[y]$.

Using the equivalent definition of $\tilde{d}$, the symmetry of $\tilde{d}$ follows directly from the symmetry of d. The triangle inequality remains. Let x, y and z be points in M, defining orbits $G[x], G[y], G[z] \subseteq M$. Choose $h, k \in G$ for which $d(h[x], y) = \tilde{d}(G[x], G[y])$ and $d(k[y], z) = \tilde{d}(G[y], G[z])$. Now we derive

$$\begin{aligned}
\tilde{d}(G[x], G[z]) &\leq d(kh(x), z) \\
&= d(h(x), k^{-1}(z)) \\
&\leq d(h(x), y) + d(y, k^{-1}(z)) \\
&= d(h(x), y) + d(k(y), z) \\
&= \tilde{d}(G[x], G[y]) + \tilde{d}(G[y], G[z]).
\end{aligned}$$

□

This theorem shows that for a metric d, which is defined on the collection of patterns $\mathcal{P}$ and which is invariant for the transformation group G, the function

$$\tilde{d}(G[A], G[B]) = \min\{\, d(g[A], B) \mid g \in G \,\}$$

defines a metric on the family of shapes $\mathcal{P}/G$. Of course, we must not forget that $(\mathcal{P}, d)$ and G need to be such that the minimum actually exists. In [6], a result similar to the previous theorem in the setting of invariant subsets of transformation groups can be found. Rucklidge [18] proved a special case of this theorem for the Hausdorff distance under Euclidean isometries.

Minimisation of a well behaved invariant metric on patterns gives a well behaved metric on shapes. This follows from the fact that the canonical mapping $A \mapsto G[A]$ from the collection of patterns $\mathcal{P}$ to the family of shapes $\mathcal{P}/G$ is continuous. Metrics on shapes are useful for object recognition. Using them, the resemblence between an input pattern and a model pattern can be determined, independent of transformation.

2.3 Properties for similarity measures

Suppose we have a transformation group G, acting on a collection of patterns $\mathcal{P}$. For reliable pattern matching, we need a similarity measure d on the collection of patterns satisfying at least the conditions stated below.

1. The function d must satisfy the metric properties. Definiteness: identical patterns should have distance zero and different patterns must have value other than zero. That is, $d(A, B) = 0$ if and only if $A = B$. Nonnegativity: identical patterns should have the lowest possible value of the similarity measure. Since the similarity of identical patterns is zero, the similarity of two different patterns should be positive. Symmetry: if A is similar to B, then B is just as similar to A, giving that $d(A, B)$ equals $d(B, A)$. Triangle inequality: if A is very similar to B and B is very similar to C, then A and C should be similar, meaning that $d(A, C)$ should not be greater than the sum of $d(A, B)$ and $d(B, C)$.
2. The measure must be invariant for G. This implies that $d(g[A], B)$ equals $d(A, g^{-1}[B])$, meaning that for all transformations $g \in G$ the image $g[A]$ is as similar to B as A is to the inverse image $g^{-1}[B]$. In addition, an invariant similarity measure can be used to define a metric on shapes using minimisation.
3. The similarity measure d must be perturbation robust: very "small" continuous transformations applied to a pattern A, resulting in a pattern B, must result in a value $d(A, B)$ that is almost zero.
4. The similarity measure d must be occlusion robust: Addition of small regions to a pattern A or removal of small regions from a pattern A, resulting in a pattern B, should give a value of $d(A, B)$ that is almost zero.

2.4 Examples of similarity measures

In the computational geometry literature, various pattern matching techniques are based on the minimisation of a similarity measure under geometric transformations. These include algorithms based on the Fréchet distance, the Hausdorff distance, and the symmetric difference. In some other techniques, like for example exact pattern matching, and bottleneck matching, the similarity measure is more or less hidden in the problem statement. Below we give a short summary of similarity measures important for geometric pattern matching. The similarity measure proposed in this paper is presented as the last one.

2.4.1 The discrete metric

Exact pattern matching is finding transformations which minimise what we will call the *discrete metric* defined for arbitrary subsets A and B of R^d by

$$d_{\mathcal{D}}(A,B) = \begin{cases} 0 & \text{if } A = B \\ 1 & \text{otherwise.} \end{cases}$$

Clearly, the discrete metric is invariant under any transformation group acting on R^d. However, it is not robust for perturbations or occlusion; even changing a single point of a pattern has the maximal possible effect on the value of the discrete metric.

2.4.2 The bottleneck distance

Bottleneck matching is based on the minimisation of what we will call the *bottleneck distance*. Given a bijection f between finite subsets A and B of R^d, define

$$m(f) = \max\{\, \|f(a) - a\| \mid a \in A \,\}.$$

Then, the bottleneck distance of two patterns A and B is given by

$$d_{\mathcal{B}}(A,B) = \min\{\, m(f) \mid f \in F(A,B) \,\},$$

where $F(A,B)$ denotes the set of all bijections from A to B. By definition, the bottleneck is invariant under isometries. It can be shown that the bottleneck distance is a metric using Theorem 1. In addition, the bottleneck distance is perturbation robust; small movements of points in patterns will not change the bottleneck distance very much. However, even slight occlusion, can result in a large increase of the bottleneck distance.

2.4.3 The Fréchet distance

Consider the collection of all sets in R^2 whose boundary is a simple closed curve. Define a metric ρ on continuous functions f and f' from the unit circle S^1 to the plane R^2 by

$$\rho(f,f') = \max\{\, \|f(x) - f'(x)\| \mid x \in \mathrm{S}^1 \,\}.$$

Define F_A as the collection of all homeomorphism from S^1 to the boundary of A. Then the *Fréchet distance* (for closed curves) is given by

$$d_{\mathcal{F}}(A,B) = \inf\{\, \rho(f,f') \mid f \in F_A \text{ and } f' \in F_B \,\}.$$

The Fréchet distance is a semimetric invariant under isometries. It is relatively robust for small deformations of patterns, but it does not allow for occlusion.

2.4.4 The Hausdorff distance

Let $\mathcal{C}$ denote the collection of all non-empty compact subsets of R^d. For compact subsets $A \subseteq \mathrm{R}^d$, define a surface $f_A : \mathrm{R}^d \to \mathrm{R}$ by

$$f_A(x) = \min\{\, \|x-y\| \mid y \in A \,\}.$$

The *directed Hausdorff distance* of A and B in $\mathcal{C}$ is

$$\vec{d}_{\mathcal{H}}(A,B) = \max\{\, f_B(a) \mid a \in A \,\}.$$

The *Hausdorff distance* on the collection $\mathcal{C}$ is given by

$$d_{\mathcal{H}}(A,B) = \max\{\, \vec{d}_{\mathcal{H}}(A,B), \vec{d}_{\mathcal{H}}(B,A) \,\}.$$

By definition, the Hausdorff distance is invariant under isometries. It can be shown that the Hausdorff distance is a metric. In addition, it is perturbation robust. However, the Hausdorff distance is not robust in case of occlusion; adding a single point to a pattern can cause a big difference in value of the Hausdorff distance.

The *partial* Hausdorff distance, defined on finite point sets, has some built-in tolerance for occlusion. It selects the k-th largest distance over points in A to the closest point in B, and the l-th largest distance over points in B to the closest point in the other direction and takes the maximum of these two distances. The main drawback of the partial Hausdorff distance is that the metric properties of definiteness, symmetry, and the triangle inequality do not hold. Furthermore, the partial Hausdorff distance depends on thresholds k and l estimating the amount of occlusion, and is relatively insensitive for changes "within the thresholds".

2.4.5 The symmetric difference

Define the solid sets $\mathcal{S}$ to be the collection consisting of all non-empty compact sets A that are equal to the closure of their interior in R^d, i.e. $\mathrm{cl}(\mathrm{int}(A)) = A$. The *symmetric difference* of two solid sets A and B in $\mathcal{S}$ is defined as

$$d_{\mathcal{S}}(A,B) = \mathrm{vol}(A - B \cup B - A),$$

where the volume of a solid set $A \in \mathcal{S}$ equals

$$\mathrm{vol}(A) = \int_{\mathrm{R}^d} \chi_A(x)\, dx.$$

Here χ_A denotes the characteristic function of A. The symmetric difference is a metric invariant under the special affine transformations $\mathrm{SAff}(d)$. It is robust for both perturbation and occlusion. Small continuous transformations of a pattern, or the addition or removal of small regions, does not have a significant effect on the symmetric difference.

2.4.6 The absolute difference

Consider the vector space $\mathbf{I}(\mathrm{R}^d)$ of real valued Riemann integrable functions on R^d, with pointwise scalar multiplication and vector addition. Let the seminorm $\|\cdot\|$ on integrable functions $\mathbf{a} \in \mathbf{I}(\mathrm{R}^d)$ be given by

$$\|\mathbf{a}\| = \int_{\mathrm{R}^d} |\mathbf{a}(x)|\, dx.$$

For solid sets $A \in \mathcal{S}$, construct the corresponding function $\mathbf{s}_A \in \mathbf{I}(\mathrm{R}^d)$ by

$$\mathbf{s}_A(x) = \mathrm{vol}(A)^{-1} \chi_A(x).$$

Now, the *absolute difference* of A and B in $\mathcal{S}$ is defined by

$$d_{\mathcal{A}}(A, B) = \|\mathbf{s}_A - \mathbf{s}_B\|.$$

Like the symmetric difference, the absolute difference is a metric robust for both perturbation and occlusion. In addition, it is invariant under the group of full affine transformations instead of just the special affine transformations.

2.5 Properties of the absolute difference

In this section, we show that the absolute difference is an affine invariant metric. First, we present a theorem which states that the absolute difference is a metric. After that, we show the absolute difference is invariant under full affine transformations.

Theorem 2 *The absolute difference is a metric on solid sets.*

Proof. The seminorm $\|\cdot\|$ on the collection of Riemann integrable functions $\mathbf{I}(\mathrm{R}^d)$ induces a semimetric ρ on functions $\mathbf{a}$ and $\mathbf{b}$ in $\mathbf{I}(\mathrm{R}^d)$ by $\rho(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|$. Let $f : \mathcal{S} \to \mathbf{I}(\mathrm{R}^d)$ be the injection $f(A) = \mathbf{s}_A$, mapping each solid set $A \in \mathcal{S}$ to the integrable function $\mathbf{s}_A \in \mathbf{I}(\mathrm{R}^d)$ as defined in Section 2.4.6. The absolute difference can be written in terms of the function f and the semimetric ρ by $d_{\mathcal{A}}(A, B) = \rho(f(A), f(B))$. It follows that $d_{\mathcal{A}}$ is a semimetric.

This leaves us with proving that $d_{\mathcal{A}}(A, B) = 0$ implies $A = B$ for all $A, B \in \mathcal{S}$. We will do this part by showing that $A \neq B$ implies $d_{\mathcal{A}}(A, B) > 0$. Without loss of generality assume that A is not a subset of B. We claim in that case that $U = \mathrm{int}(A) - B$ is not empty. Suppose U were empty, then clearly $\mathrm{int}(A) \subseteq B$. This implies $\mathrm{cl}(\mathrm{int}(A)) \subseteq \mathrm{cl}(B)$. But since A and B are solid sets, the latter is equivalent to $A \subseteq B$, which gives a contradiction. Since $U \subseteq A - B$ is a

non-empty open subset of R^d, it contains some open hypercube with nonzero volume $\epsilon > 0$. Now we derive

$$\begin{aligned} d_{\mathcal{A}}(A, B) &\geq \int_U |\mathbf{s}_A(x) - \mathbf{s}_B(x)|\, dx \\ &= \int_U \mathrm{vol}(A)^{-1}\, dx \\ &\geq \mathrm{vol}(A)^{-1} \cdot \epsilon \\ &> 0. \end{aligned}$$

□

We proceed by showing that the absolute difference is invariant under the group of full affine transformations. First, Lemma 2 gives the action of C^1 diffeomorphisms on the space of integrable functions and shows that this results in a transformation group. Then, Lemma 3 shows that the seminorm defined on the integrable functions is invariant under the this action. Finally, Theorem 3 uses Lemmas 2 and 3 to prove that the absolute difference is invariant under affine transformations.

The following lemma states that defining a particular action of the C^1 diffeomorphisms for real valued functions on R^d, gives a transformation group.

Lemma 2 *Let $g : \mathrm{R}^d \to \mathrm{R}^d$ be a C^1 diffeomorphism and let $\mathbf{a} \in \mathbf{I}(\mathrm{R}^d)$. Let $r_g(y) = |\det(D_g(y))|^{-1}$ be the reciprocal of the absolute Jacobi determinant of g in y. Now g and $\mathbf{a}$ define a function $g\langle\mathbf{a}\rangle \in \mathbf{I}(\mathrm{R}^d)$ by $g\langle\mathbf{a}\rangle(x) = r_g(g^{-1}(x))\, \mathbf{a}(g^{-1}(x))$. Under this action, the C^1 diffeomorphisms form a transformation group for the integrable functions $\mathbf{I}(\mathrm{R}^d)$.*

The proof of this lemma can be found in appendix A.

The full affine transformations form a subgroup of the C^1 diffeomorphisms and therefore are a transformation group for $\mathbf{I}(\mathrm{R}^d)$. Note that for affine transformations g, the value of $r_g(y)$ is constant in $y \in \mathrm{R}^d$. This is because $D_g(y)$ can be seen as the linear component of an affine transformation g, where $r_g(y)$ is the reciprocal of the absolute valued determinant of this linear transformation. The following lemma shows that the seminorm defined on the integrable functions is invariant under the proposed action of C^1 diffeomorphisms.

Lemma 3 *The seminorm $\|\cdot\|$ on the integrable functions $\mathbf{I}(\mathrm{R}^d)$ is invariant for the C^1 diffeomorphisms.*

For the proof, we refer to appendix B. Finally, we arrive at the theorem stating invariance.

Theorem 3 *The absolute difference is invariant for affine transformations.*

Proof. By Lemma 3, the seminorm $\|\cdot\|$ on $\mathbf{I}(\mathrm{R}^d)$ is invariant under full affine transformations. Because the mapping $\mathbf{a} \mapsto g\langle\mathbf{a}\rangle$ is linear, it follows that the semimetric $\rho(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|$ is also invariant under full affine transformations.

The mapping $f : A \mapsto \mathbf{s}_A$ from the solid functions $\mathcal{S}$ to the integrable functions $\mathbf{I}(\mathrm{R}^d)$ is equivariant for affine transformations $g : \mathrm{R}^d \to \mathrm{R}^d$:

$$\begin{aligned}\mathbf{s}_{g[A]}(x) &= \mathrm{vol}(g[A])^{-1}\chi_{g[A]}(x) \\ &= |\det(D_g(0))|^{-1}\chi_A(g^{-1}(x)) \\ &= g\langle \mathbf{s}_A \rangle(x).\end{aligned}$$

The absolute difference of two solid sets A and B equals $\rho(f(A), f(B))$. The invariance of ρ and the equivariance of f together show that the absolute difference is invariant under full affine transformations. $\square$

3 Minimisation of the similarity measure

Now, we explain our hierarchical pattern matching algorithm. The algorithm recursively subdivides transformation space, computing a transformation which brings the similarity measure within any chosen accuracy of the global minimum. The algorithm can be applied to the Hausdorff distance, the symmetric difference, and the absolute difference.

Section 3.1 gives the general pattern matching algorithm. Section 3.2 derives inequalities which can be used to bound both the Hausdorff distance and the absolute difference. After that, Section 3.3 gives expressions of bounds for the absolute difference. Finally Section 3.4 gives algorithms that compute the these bounds.

3.1 The basic algorithm

We address the problem of finding a transformation g minimising the value of some similarity measure $d(g[A], B)$, given two fixed patterns A and B. Let H be a subset of some transformation space G. Consider the similarity between B and each image $h[A]$, for all $h \in H$. This set of values has an infimum

$$m(H) = \inf\{d(h[A], B) \mid h \in H\}.$$

The algorithm is based on the existence of a lower bound $l(H)$ for $m(H)$. During computation, we maintain a finite collection of compact sets H. In each iteration, we pick the H from the collection that has the smallest value of $l(H)$. Subsequently, we split H up into smaller sets H' and H'' which will in turn be added to the collection. After this the next iteration begins. The process continues until we find a set H that is "accurate enough". The following lemma shows how we may find transformations that approximate the minimum, using a collection of compact subsets of G.

Lemma 4 *Let l and r be real valued functions on compact $K \subseteq G$ satisfying*

$$l(K) \leq d(k[A], B) \leq l(K) + r(K)$$

for all $k \in K$. Suppose $\mathcal{H}$ is a finite collection of compact sets with $m(\bigcup \mathcal{H}) = m(G)$. If $H \in \mathcal{H}$ is such that $l(H) = \min\{\, l(K) \mid K \in \mathcal{K} \,\}$, then for all $h \in H$,

$$m(G) \leq d(h[A], B) \leq m(G) + r(H).$$

Proof. The inequality $m(G) \leq d(h[A], B)$ holds by definition of m. Since $m(\bigcup \mathcal{H}) = m(G)$,

$$\begin{aligned} l(H) &\leq m(H) \\ &= \min\{\, m(K) \mid K \in \mathcal{H} \,\} \\ &= m(\bigcup \mathcal{H}) \\ &= m(G). \end{aligned}$$

Adding $r(H)$ to both sides of the inequality gives

$$d(h[A], B) \leq l(H) + r(H) \leq m(G) + r(H).$$

□

This lemma provides the basic idea for an hierarchical approximation algorithm which can be used to find a transformation $h \in G$ satisfying

$$m(G) \leq d(h[A], B) < m(G) + \epsilon$$

for any fixed accuracy $\epsilon > 0$. Start with a compact subset $K \subseteq G$ satisfying $m(K) = m(G)$. We cover this K with a finite collection of compact sets $\mathcal{H}_1$. During iteration k of the algorithm we consider the finite covering $\mathcal{H}_k$, starting with $\mathcal{H}_1$. We select a set $H \in \mathcal{H}_k$ with a minimal value of l. If $r(H) < \epsilon$, each transformation $h \in H$ approximates the minimum with the desired accuracy. Otherwise, we split the compact H into two "smaller" compact sets H' and H'' covering H. Then we proceed to the iteration $k+1$ using $\mathcal{H}_{k+1} = \mathcal{H}_k \cup \{\, H', H'' \,\} - \{\, H \,\}$ which is still a compact covering of K.

Now we investigate how to split up a compact set in transformation space into a pair of "smaller" ones. For this purpose, we restrict our discussion to transformation groups that can be represented in finite-dimensional Euclidean space. For example, we treat the full affine transformations and its subgroups as a subspaces of $\mathrm{R}^{d(d+1)}$. In the following, we write $G \subseteq \mathrm{R}^p$, assuming that elements of G can be represented using p-tuples of real parameters.

We define an p-dimensional *hyperblock* in R^p as a finite product of closed intervals $C = C_1 \times \cdots \times C_p$. The i-th *projection mapping* π_i is the function

$$\pi_i : (x_1, \ldots, x_p) \to x_i : \mathrm{R}^p \to \mathrm{R}.$$

A hyperblock C can be defined uniquely by equating, for each $i = 1, \ldots, p$, the image of C under the i-th projection mapping $\pi_i(C)$ with a closed interval C_i. If we consider the Euclidean space R^p containing the transformation group G, we call each p-dimensional hyperblock $C \subseteq \mathrm{R}^p$ a *cell*. Each cell $C \subseteq \mathrm{R}^p$ determines a set of transformations given by the intersection $C \cap G$.

```
CellSplit(C_1 × ··· × C_p)
 1  Choose the lowest index j such that C_j has maximal length.
 2  for j ← 1 to p
 3  do if i = j
 4        then s ← (1/2)(min C_i + max C_i)
 5             C'_i ← [min C_i, s]
 6             C''_i ← [s, max C_i]
 7        else C'_i ← C_i
 8             C''_i ← C_i
 9  switch
10    case C' ∩ G ≠ ∅ and C'' ∩ G ≠ ∅ :
11         return C' and C''.
12    case C' ∩ G = ∅ :
13         return CellSplit(C'')
14    case C'' ∩ G = ∅ :
15         return CellSplit(C')
```

Algorithm 1: The cell-splitting procedure.

Algorithm 1 describes the splitting procedure that will be used in our main matching algorithm. For a cell C with $\operatorname{diam}(C \cap G) > 0$, this algorithm terminates, returning cells C' and C'' having strictly smaller diameters than the original cell. Cells with arbitrary small diameter can be produced by repeating the splitting procedure a finite number of times. Clearly, the cells C' and C'' computed by Algorithm 1 satisfy $C \cap G = (C' \cup C'') \cap G$. Furthermore, if we start with a cell C for which $C \cap G$ is compact in G, the resulting cells C' and C'' have intersections $C' \cap G$ and $C'' \cap G$, both compact in G.

At this point, we have the requisites to write down an algorithm which can be used to approximate the minimum value of the metric under transformation, see Algorithm 2. Lemma 4 tells us that at the start of each iteration the set of transformations $H = C \cap G$ is such that $d(h[A], B) \leq m(G) + r(H)$ for all $h \in H$. If the guard condition in line 5 is satisfied, meaning $r(H) < \epsilon$, then for all $h \in H$,

$$m(G) \leq d(h[A], B) < m(G) + \epsilon.$$

An important problem, which we have not addressed yet, is the termination of the algorithm. The value $r(H)$ must eventually become strictly smaller than ϵ, otherwise the algorithm will not terminate. The following lemma gives a condition, similar to the notion of uniform continuity, under which the algorithm terminates.

Lemma 5 *Let G be a subset of R^p. Suppose for all compact $K \subseteq G$ and $\epsilon > 0$, there is a number $\delta > 0$ such that for all compact $C \subseteq \mathrm{R}^d$ with $\operatorname{diam}(C) < \delta$, and $C \cap G \subseteq K$, the inequality $r(C \cap G) < \epsilon$ holds. Then Algorithm 2 terminates.*

`MinimumApproximate`(ϵ)

1 Determine a compact $K \subseteq G$ satisfying $m(K) = m(G)$.
2 Cover K in R^p with a finite collection of cells $\mathcal{C}_1$.
3 Insert all elements of $\mathcal{C}_1$ in a priority queue `Q`.
4 Select a cell C with minimal value of $l(C \cap G)$ in `Q`.
5 **while** $r(C \cap G) \geq \epsilon$
6 **do** Compute cells C' and C'' using `CellSplit`(C).
7 Insert C' and C'' into `Q`.
8 Remove C from `Q`.
9 Select a cell C with minimal value of $l(C \cap G)$ in `Q`.
10 **return** an arbitrary transformation $h \in C \cap G$.

Algorithm 2: Finding an almost minimal transformation.

Proof. First, we show $r(C \cap G) > 0$ implies $\mathrm{diam}(C \cap G) > 0$, making the call to the cell splitting procedure in line 6 of Algorithm 2 correct. Suppose $\mathrm{diam}(C \cap G) = 0$. Then, for any $\epsilon > 0$, we can choose a cell C' such that $C' \cap G = C \cap G$ satisfying $r(C' \cap G) < \epsilon$. This implies that $r(C \cap G)$ must be zero.

Let C_i be the cell C at the start of iteration i of Algorithm 2 (line 5). Suppose the algorithm does not terminate for some $\epsilon > 0$. Then $C_1, \ldots, C_i, \ldots$ is an infinite sequence of cells. This sequence must contain an infinite nested subsequence of cells

$$C_{i_1} \supseteq \ldots \supseteq C_{i_k} \supseteq \ldots$$

Since Algorithm 1 splits the cells on their longest interval, $\mathrm{diam}(C_{i_{k+1}}/C_{i_k}) \leq (1 - 3/(4p))^{0.5}$. But then, $\mathrm{diam}(C_{i_k}) < \delta$ for some k, implying $r(C_{i_k} \cap G) < \epsilon$, meaning the algorithm would have terminated at the start of iteration k. □

Finally, we conclude that our approximation algorithm is correct provided l and r satisfy the conditions stated in the previous two lemmas:

Theorem 4 *Suppose l and r satisfy the conditions of Lemmas 4 and 5. Then for any $\epsilon > 0$, Algorithm 2 computes an $h \in G$ satisfying*

$$m(G) \leq d(h[A], B) \leq m(G) + \epsilon.$$

3.2 Deriving bounding functions

In this section, we provide bounds for the Hausdorff distance, and the absolute difference as defined in Section 2.4. This results in functions l and r which can be used in Algorithm 2 to find matches. In the following, we assume patterns A and B and a compact set of transformations H are fixed.

Consider the Hausdorff distance $d = d_{\mathcal{H}}$ defined on the collection of non-empty compact sets $\mathcal{C}$, under Euclidean isometries $G = \mathrm{Iso}(d)$. We derive a lower bound l and an accuracy r for d in the sense of Lemma 4.

We bound Hausdorff distance over a set of isometries H by introducing a special construct. The *traced volume* of A is defined by $T_A = \bigcup_{h \in H} h[A]$. The traced volume of B is defined by $T_B = \bigcup_{h \in H} h^{-1}[B]$, using the inverse set of transformations. The Hausdorff distance under transformation can be written in terms of the directed Hausdorff distance:

$$d_{\mathcal{H}}(g[A], B) = \max\{\, \vec{d}_{\mathcal{H}}(A, g^{-1}[B]), \vec{d}_{\mathcal{H}}(B, g[A]) \,\}.$$

Substituting the traced volumes into this equation gives the following lower bound l, upper bound u and accuracy r for the Hausdorff distance:

$$l(H) = \vec{d}_{\mathcal{H}}(A, T_B) + \vec{d}_{\mathcal{H}}(B, T_A),$$

$$u(H) = \vec{d}_{\mathcal{H}}(T_A, B) + \vec{d}_{\mathcal{H}}(T_B, A),$$

$$r(H) = u(H) - l(H).$$

Now consider the absolute difference $d = d_{\mathcal{A}}$ defined on the collection of solid sets $\mathcal{S}$ under the full affine transformations. The *traced envelope* for $A \subseteq \mathrm{R}^d$ is the function defined for $x \in \mathrm{R}^d$ by $\mathbf{t}_A(x) = \max\{\, h\langle \mathbf{s}_A \rangle(x) \mid h \in H \,\}$. We write the traced envelope for B analogous using $h^{-1}\langle \mathbf{s}_B \rangle(x)$ instead of $h\langle \mathbf{s}_A \rangle(x)$. Define a "directed" distance $\vec{\rho}$ on integrable functions $\mathbf{a}, \mathbf{b} \in \mathbf{I}(\mathrm{R}^d)$ by

$$\vec{\rho}(\mathbf{a}, \mathbf{b}) = \int_{\mathrm{R}^d} \max\{\, 0, \mathbf{a}(x) - \mathbf{b}(x) \,\}\, dx.$$

We express the absolute difference symmetrically using the directed distance:

$$d_{\mathcal{A}}(A, B) = \vec{\rho}(\mathbf{s}_A, \mathbf{s}_B) + \vec{\rho}(\mathbf{s}_B, \mathbf{s}_A).$$

Since $\vec{\rho}$ is invariant under the action of full affine transformations,

$$d_{\mathcal{A}}(g[A], B) = \vec{\rho}(\mathbf{s}_A, g^{-1}\langle \mathbf{s}_B \rangle) + \vec{\rho}(\mathbf{s}_B, g\langle \mathbf{s}_A \rangle).$$

Each transformed $h\langle \mathbf{s}_A \rangle$ is bounded pointwise from above by the traced envelope $\mathbf{t}_A$, for $h \in H$. Similarly $h^{-1}\langle \mathbf{s}_B \rangle$ is bounded from above by $\mathbf{t}_B$. Therefore, a lower bound l, an upper bound u, and an accuracy r for the absolute difference are given by the following equations,

$$l(H) = \vec{\rho}(\mathbf{s}_A, \mathbf{t}_B) + \vec{\rho}(\mathbf{s}_B, \mathbf{t}_A),$$

$$u(H) = \vec{\rho}(\mathbf{t}_A, \mathbf{s}_B) + \vec{\rho}(\mathbf{t}_B, \mathbf{s}_A),$$

$$r(H) = u(H) - l(H).$$

3.3 Bounds for sets of hyperblocks

We now focus on the problem of actually computing a simple lower bound l for the absolute difference between finite unions of hyperblocks. The computation of the upper bound u, allowing computation of the accuracy r, can be done in a similar fashion. Since cells are not allowed to contain singular transformations, we will write C instead of $C \cap G$, in the following. First, we show how to represent affine transformations. Then, we compute hyperblocks that contain traced volumes. Using these hyperblocks, we compute functions that are defined constant on the hyperblocks and zero otherwise, and bound the traced envelopes from above.

First, we represent the full affine transformations acting on R^d in the finite-dimensional vector space R^p, where $p = d(d+1)$. The affine transformations can be seen as compositions $t \circ l$ of full linear transformations l, and translations t. The first d indices of R^p are used to represent a translation vector, while the last d^2 indices represent a matrix expressing a linear transformation. Thus an affine transformation g can be written as an element of R^p by

$$g = (t_1, \ldots, t_d) \times (l_{11}, \ldots l_{1d}) \times \cdots \times (l_{d1}, \ldots l_{dd}).$$

The d elements t_i form a translation vector. The square matrix having the element l_{ij} at row i and column j, represents the linear transformation. A cell of affine transformations, being a hyperblock in R^p, can be written as the product set of closed intervals

$$C = \prod_{i=1}^{d} T_i \times \prod_{j=1}^{d} L_j^{(1)} \times \cdots \times \prod_{j=1}^{d} L_j^{(d)}$$

Write a hyperblock N in R^d as a product of d intervals $\prod_{i=1}^{d} N_i$. We compute the i-th interval of a hyperblock $V(C, N)$ by

$$\pi_i(V(C,N)) = \{\, t_i + \sum_{j=1}^{d} l_{ij} x_j \mid t_i \in T_i, l_{ij} \in L_j^{(i)}, x_j \in N_i \,\}.$$

This hyperblock contains the union of all $h[N]$ under transformations $h \in C$. Similarly, we define a hyperblock $V'(C, N)$ which uses elements of the inverse matrix instead of l_{ij}. This makes the computation a bit more complicated. However, using Cramer's rule, the elements of the inverse matrix are expressed as polynomials in the original matrix elements, and can therefore be bounded quite easy.

Suppose A and B can be written as disjoint unions of d-dimensional hyperblocks $\bigcup_{i=1}^{m} A_i$ and $\bigcup_{j=1}^{n} B_j$, respectively. Abbreviate $V(C, A_i)$ by V_i for $i = 1, \ldots, m$. Similarly, write V_j' instead of $V'(C, B_j)$ for $j = 1, \ldots, n$. We obtain valid bounds l, u and r for the Hausdorff distance $d_{\mathcal{H}}$ by replacing T_A and T_B, used in the previous section, with $\bigcup_{i=1}^{m} V_i$ and $\bigcup_{i=1}^{n} V_j'$, respectively.

We proceed with bounding the traced envelopes, giving simple bounds for the absolute difference. The maximum height of the function $\mathbf{s}_A$ under all affine transformations in a cell C is given by

$$q_A = \mathrm{vol}(A)^{-1} \max\{\, |\det(L)|^{-1} \mid l_{ij} \in L_j^{(i)} \,\},$$

where L is the square matrix having element l_{ij} at row i and column j. For each hyperblock A_i, define the function $\mathbf{v}_i : \mathrm{R}^d \to \mathrm{R}$ by

$$\mathbf{v}_i(x) = \begin{cases} q_A & \text{if } x \in V_i \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, the pointwise maximum over these functions, denoted by $\max \mathbf{v}_i$, is an upper bound for the traced envelope $\mathbf{t}_A$. We may also define q_B, using $\mathrm{vol}(B)^{-1}$ and the absolute valued determinants instead of their inverses. This gives functions $\mathbf{v}'_j$ for $j = 1, \ldots, n$ whose pointwise maximum $\max \mathbf{v}'_j$ bounds the traced envelope $\mathbf{t}_B$. We get bounds l, u and r for the absolute difference by replacing the traced envelopes $\mathbf{t}_A$ and $\mathbf{t}_B$ used in the previous section by $\max \mathbf{v}_i$ and $\max \mathbf{v}'_j$, respectively.

3.4 An algorithm for computing bounds

We present simple algorithms which can be used to compute the lower bounds for the absolute distance given in the previous section. The basic data structure used in the algorithm is the kd-tree; see de Berg et al. [16]. We use a special kind of kd-tree containing information for speeding up the computation of the bound, as will be seen below.

The computation of $l(C)$ is performed in three stages. First, we build and initialise augmented kd-trees on A and B. Second, we label the kd-trees using the hyperblocks V_i and V'_j. Third, we evaluate the labelled kd-trees using the reciprocal volumes of A and B and the bounds q_A and q_B, resulting in the value of the bound $l(C)$.

For a disjoint union of hyperblocks $A_1 \cup \cdots \cup A_m$, we build a balanced kd-tree containing the hyperblocks in its leaves. Each internal node `n` in the kd-tree has child nodes `left`[`n`] and `right`[`n`]. In addition, each node `n` has a corresponding hyperblock `box`[`n`]. This hyperblock is the bounding box for all hyperblocks in the subtree rooted at `n`. For leaves `n`, we set it to A_i, and for internal nodes `n`, we set it to be the smallest hyperblock containing both `box`[`left`[`n`]] and `box`[`right`[`n`]] for an internal node. Algorithm 3 gives code for building and initialising the kd-tree.

Algorithm 4, the tree labelling procedure, works as follows. Suppose we built a kd-tree on the pattern B. For each hyperblock $A_i \subseteq A$, we compute the hyperblock V_i. If the subset of B represented by a subtree rooted at a node `n` is contained in the hyperblock V_i, we set the flag `covered`[n] to be true. In each leaf `n`, we maintain a union of hyperblocks `c`[`n`], which indicates the part of `box`[`n`] yet covered by any previously processed V_i.

```
KdTreeBuild(A_1 ∪ ··· ∪ A_m)
 1  if m = 1
 2    then Create a leaf n.
 3         box[n] ← A_1
 4         c[n] ← ∅
 5    else Create an internal node n.
 6         for j ← 1 to d
 7         do r_j ← ∞
 8            s_j ← −∞
 9            for i ← 1 to m
10            do r_j ← min{ r_j, min π_j(A_i) }
11               s_j ← max{ s_j, max π_j(A_i) }
12         box[n] ← [r_1, s_1] × ··· × [r_d, s_d]
13         Select the smallest k with maximal s_k − r_k.
14         Sort the values min π_k(A_i), for i = 1, ..., m.
15         Let (i_1, ..., i_m) be the resulting permutation.
16         t ← ⌊½m⌋
17         left[n] ← KdTreeBuild(A_{i_1} ∪ ··· ∪ A_{i_t})
18         right[n] ← KdTreeBuild(A_{i_{t+1}} ∪ ··· ∪ A_{i_m})
19  covered[n] ← false
20  return n
```

Algorithm 3: Building the kd-tree.

```
KdTreeLabel(n, V)
1  if covered[n] = false and box[n] intersects V
2    then if box[n] ⊆ V
3           then covered[n] ← true
4         if n is a leaf
5           then c[n] ← c[n] ∪ (box[n] ∩ V)
6           else KdTreeLabel(left[n], V)
7                KdTreeLabel(right[n], V)
```

Algorithm 4: Labelling the kd-tree.

```
KdTreeEval(n, r, q)
 1  if n is a leaf
 2     then return  max{ 0, r − q } · vol(c[n]) + r · vol(box[n] − c[n])
 3     else  return KdTreeEval(left[n], r, q) + KdTreeEval(right[n], r, q)
```

Algorithm 5: Evaluating the labelled tree.

```
LowerBound(C, A, B)
 1  nA ← KdTreeBuild(A)
 2  nB ← KdTreeBuild(B)
 3  for i ← 1 to m
 4  do KdTreeLabel(nB, V_i)
 5  for j ← 1 to n
 6  do KdTreeLabel(nA, V'_j)
 7  return KdTreeEval(nB, vol(B)^{-1}, q_A) + KdTreeEval(nA, vol(A)^{-1}, q_B)
```

Algorithm 6: Computing the lower bound $l(C)$ given A and B.

Algorithm 5 is used to evaluate the labelled tree resulting in the desired value of $l(C)$. The algorithm, if called for the labelled tree built on B, computes the value $\vec{\rho}(\mathbf{s}_B, \max \mathbf{v}_i)$ which is a summand of the bound $l(C)$ defined in Section 3.3. If called for the labelled tree built on A, the algorithm computes the other summand of this bound. Algorithm 5 performs a traversal of the kd-tree built on B starting with the root. If a leaf n, representing a hypercube $\mathtt{box}[\mathtt{n}] = B_j$ is reached, we return $\vec{\rho}(\mathbf{s}_{B_j}, \max \mathbf{v}_i)$. Since the hyperblocks B_j stored in the leaves are disjoint, the sum over all leaves equals $\vec{\rho}(\mathbf{s}_B, \max \mathbf{v}_i)$.

Algorithm 6 illustrates the general procedure for computing the lower bound for the absolute difference $l(C)$, for cells of affine transformations C, and patterns A and B. First kd-trees for both A and B are computed. Then the tree on B is labelled using the hyperblocks V_i. The same is done for the tree built on A using V'_j for $j = 1, \ldots, n$. Summing the results of both tree evaluations gives the desired bound $l(C)$.

Actually, it is not necessary that Algorithm 6 builds the kd-trees for both A and B. The kd-trees can be built once, and passed to the algorithm each time we want to compute the value $l(C)$ for some cell C. The only thing we have to take care of in Algorithm 6 is resetting the values of $\mathtt{c}[\mathtt{n}]$ and $\mathtt{covered}[\mathtt{n}]$ for all nodes n in both trees.

The complexity of the tree building algorithm as it is written down here is $O(m \log^2 m)$. It is possible to compute the kd-tree in $O(m \log m)$ by sorting the hyperblocks on each coordinates separately before the main call of the algorithm. When splitting up the collection of hyperblocks, the orders for the subcollections can then be maintained in linear time, omitting the need for the sorting step in line 11.

4 Results

We implemented the techniques from the previous section for both the directed Hausdorff distance and the directed absolute difference. This section contains results obtained by running a series of tests on pairs of feature patterns. For each test, feature patterns were obtained from digital images using image processing techniques. In some tests, the images are different views of the same object, while other examples consist of an image and a transformed subimage.

4.1 The tests

Figures 1–10 depict the tests. In each figure, the upper row contains the two original grey-scale images. The patterns resulting from feature detection are superimposed upon them. The white dots in the upper left subfigure form the pattern A, while the black dots in the upper right subfigure form the pattern B. The middle and lower rows contain matches. The matches on the left side in the middle and lower row were obtained by minimisation of the Hausdorff distance. The matches on the right side were obtained using the absolute difference. The middle row shows the pattern A transformed under the minimising match, superimposed on the pattern B. The lower row shows the grey-scale image corresponding to A, transformed under the same transformation, averaged with the grey-value image corresponding to B. The transformation is always found using the feature patterns A and B only, and then applied to the grey-scale images to allow visual verification. This should not be confused with image registration techniques.

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 plane design. 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 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. Figures 1–7 show edge patterns obtained by Sobel edge detection followed by thresholding. Figures 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.

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 directed 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 directed Hausdorff distance to be arbitrarily small. Since such degenerate matchings are not very interesting in a comparison, the method using the directed Hausdorff distance had its transformation space restricted to exclude arbitrary small scalings. Table 1 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 minimum match.

Table 2 shows, for each test, the transformation group which was used for matching, and the cardinalities of the patterns. The group TS(2) mentioned in the table denotes scaling combined with translation, both in two directions. Table 3 contains test statistics for both the 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 absolute difference took longer to compute, the results were accurate, while the Hausdorff distance produced "false" matches in these cases. Moreover, for the examples involving scaling (tests five to ten), the transformation space for the Hausdorff distance is only half the size of that for the absolute difference, because of the restriction on the scaling.

4.2 Discussion

Figure 1, indicates reliable performance for both the Hausdorff distance and the absolute difference. The marking is placed precisely on top of its occurrence in both cases. However, Figure 2 shows that the Hausdorff distance is more likely to be confused in areas dense with points. The mismatch in Figure 2 under the Hausdorff distance is caused by the fact that each point in A has a close point in B. The absolute difference proved more discriminative in this case and produced a visually plausible match. Figures 3 and 4 show nice results for both the Hausdorff distance and the absolute difference. In each case, the right isometry was found, matching A with a part of the three-view B. Both distances succeeded in ignoring the two "false" views in each of the three-views. Figure 5 and 6 show a slightly better matches for the absolute difference than for the Hausdorff distance. Figure 7 is the first example with a significant amount of occlusion in both patterns A and B. The picture below left clearly shows that the Hausdorff distance is "confused" by the occlusion in A and produces an unplausible match. The picture to the right of it, produced using the absolute difference, gives a visually flawless match. Figure 8 illustrates the instable behaviour of the Hausdorff distance under full affine transformations: the area of the transformed image of A is much too small. The matches produced by the absolute difference, middle and below to the right, seem much better. We encounter similar behaviour under full affine transformations in Figures 9 and 10.

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

Table 1: Initial parameters for linear transformation components.

test figure	group	# features A	# features B
1	T(2)	60	1084
2	T(2)	110	773
3	Mot(2)	189	439
4	Mot(2)	203	637
5	TS(2)	175	122
6	TS(2)	308	317
7	TS(2)	717	740
8	Aff(2)	16	17
9	Aff(2)	23	29
10	Aff(2)	20	20

Table 2: Overview of the tests.

Table 4 shows the determinants of the linear parts of the transformations computed by both implementations, for all tests involving translation and scaling or affine transformations. This 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.

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 3: 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 4: Determinants of the linear components of reported matches.

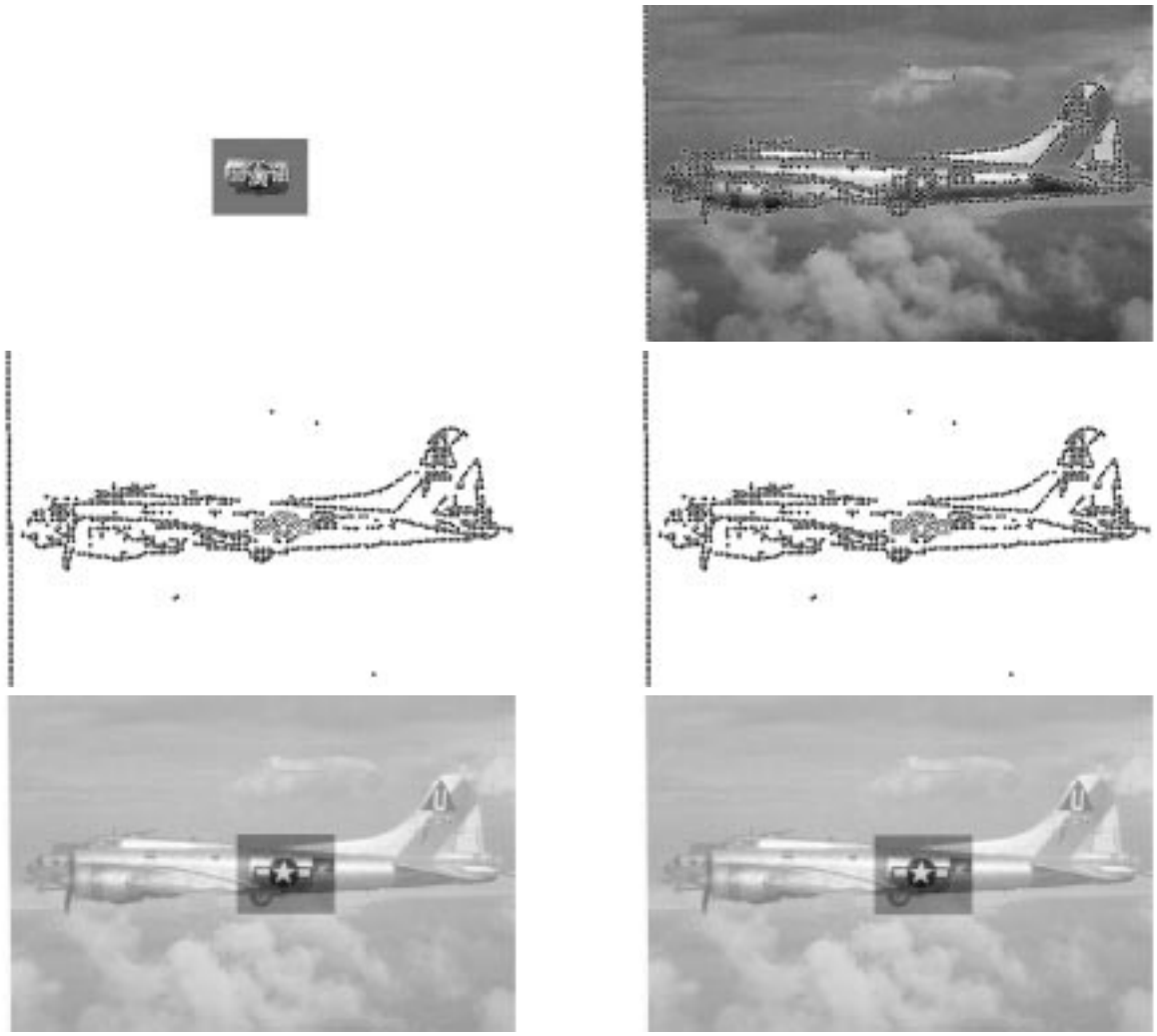

Figure 1: B17 bomber with subimage.

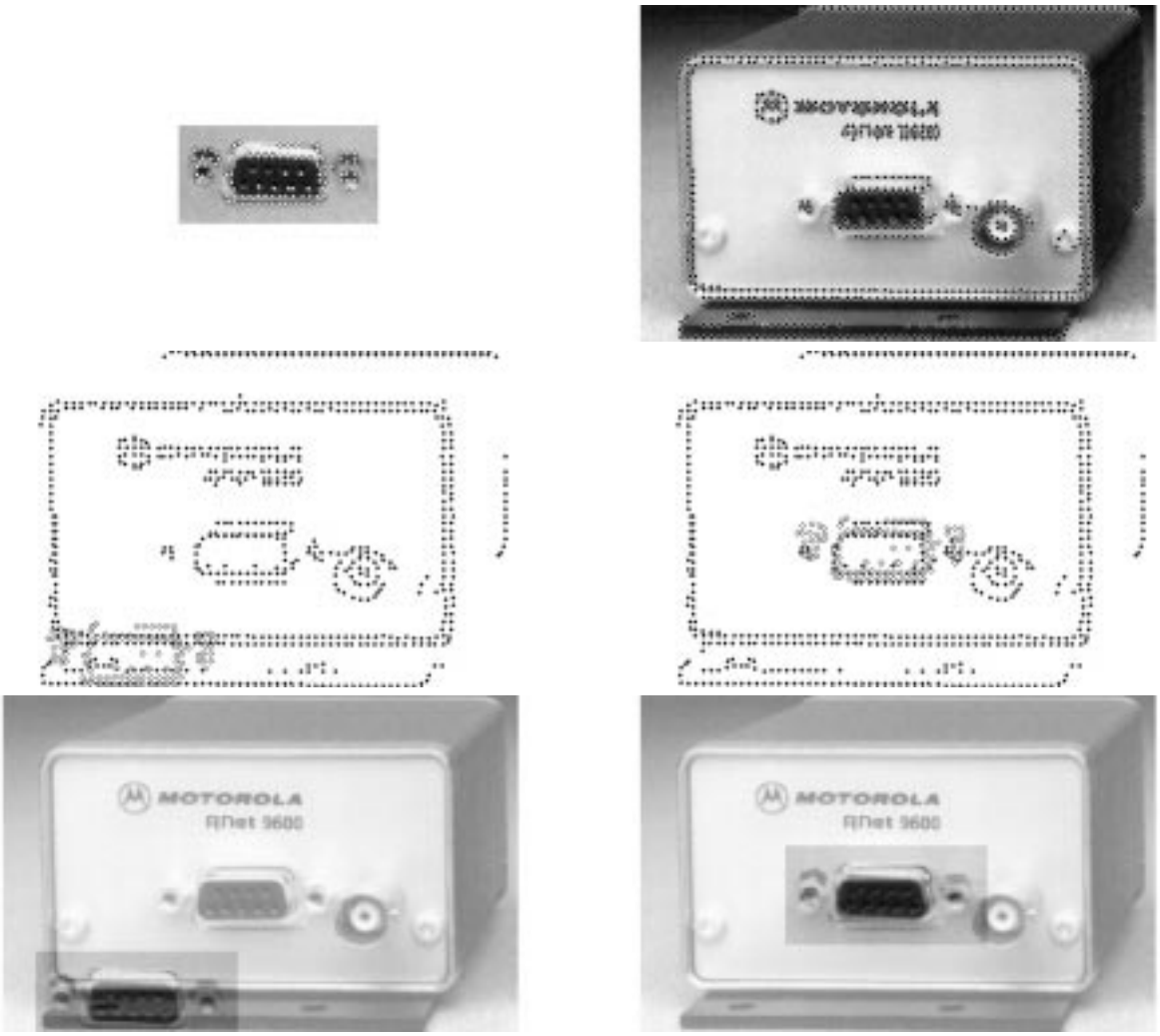

Figure 2: Device with a different connector.

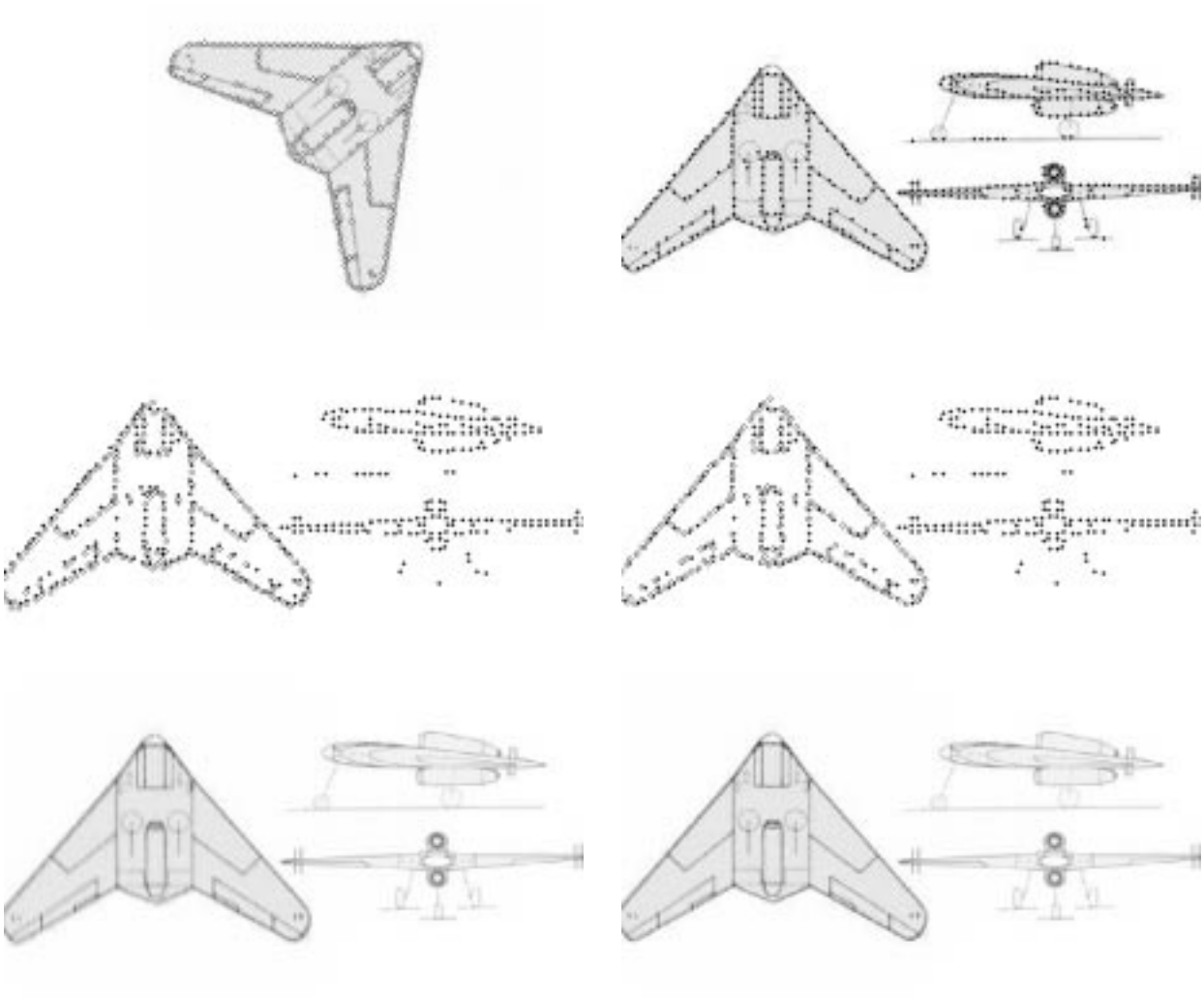

Figure 3: Three-view drawing with subimage.

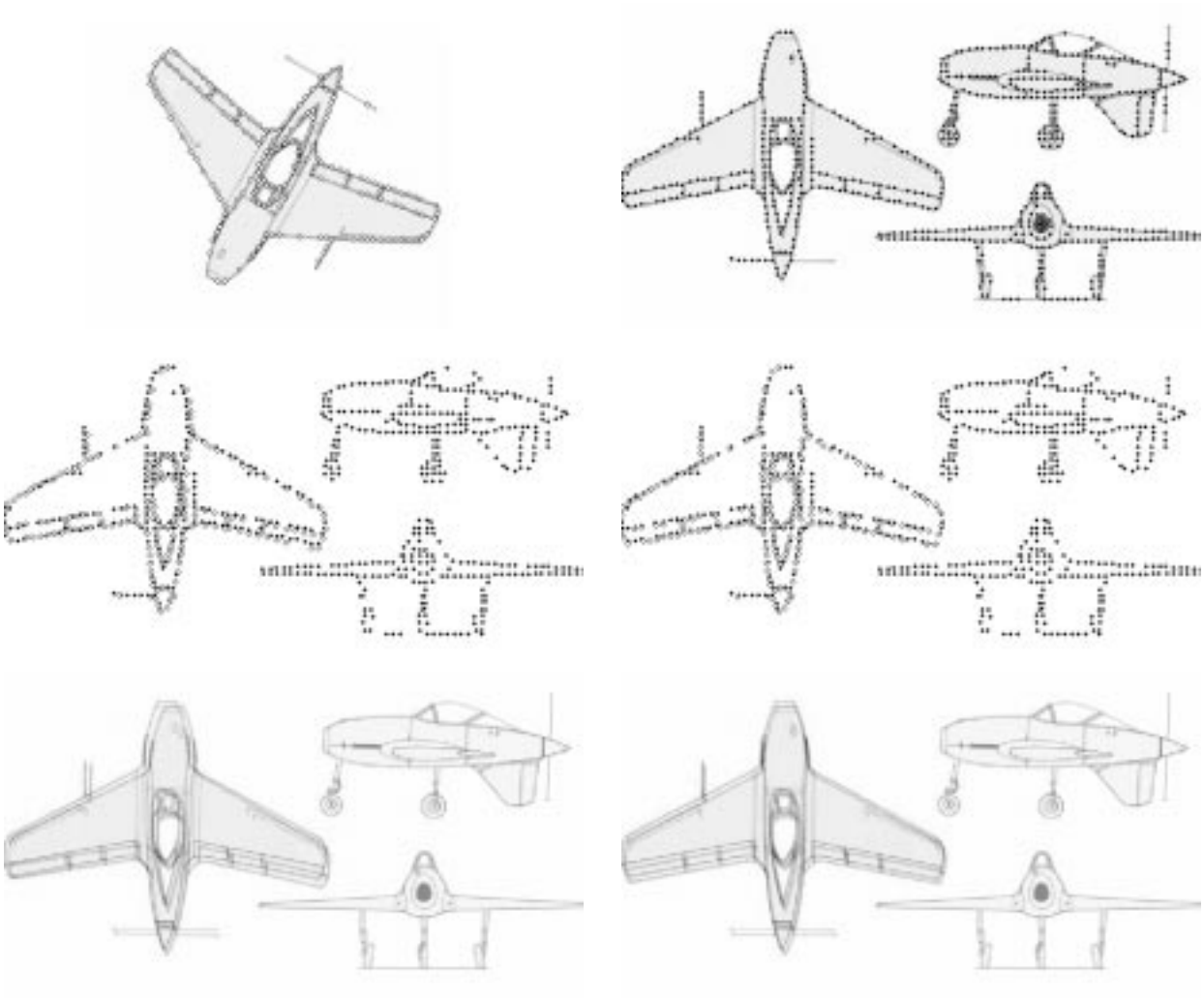

Figure 4: Three-view drawing with subimage.

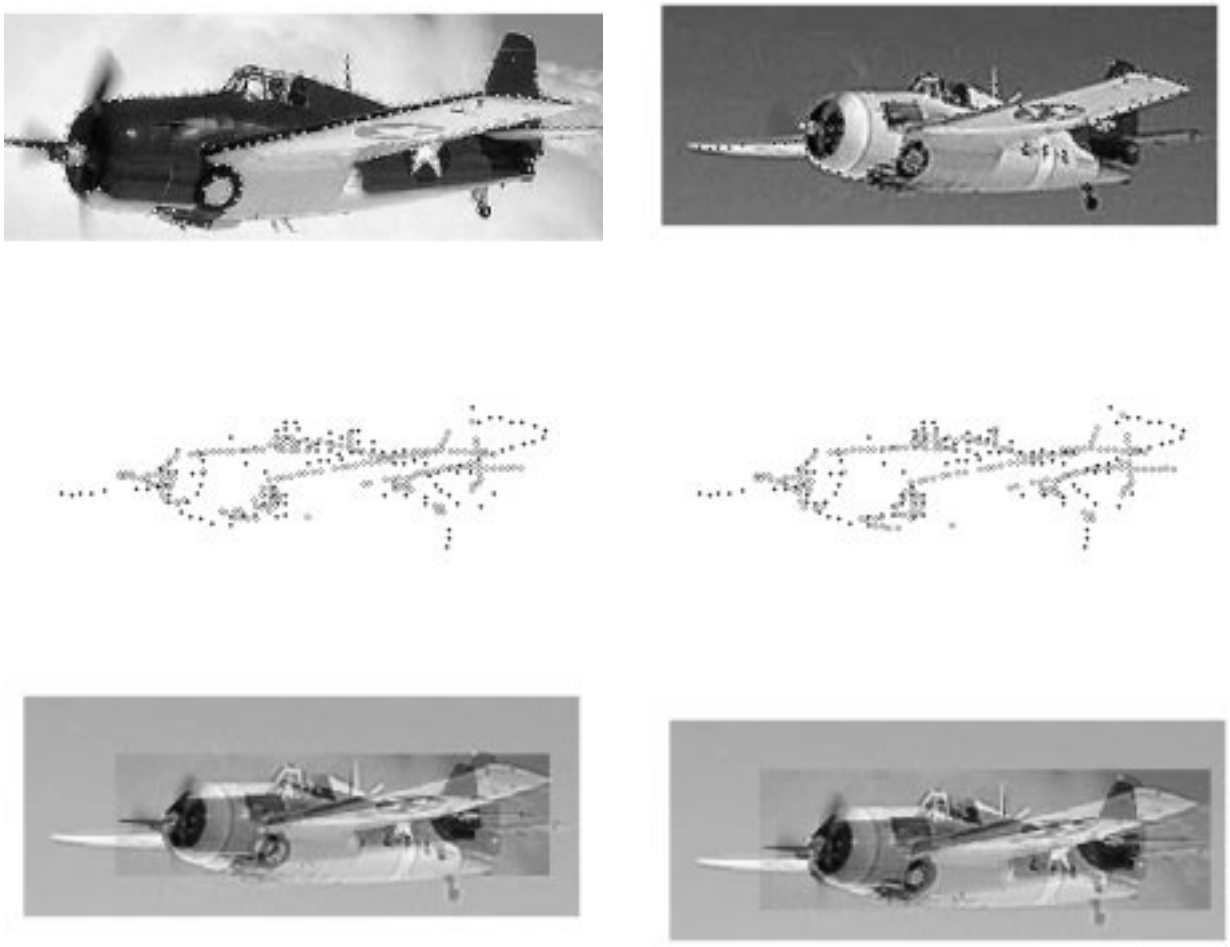

Figure 5: Two different planes of the same type.

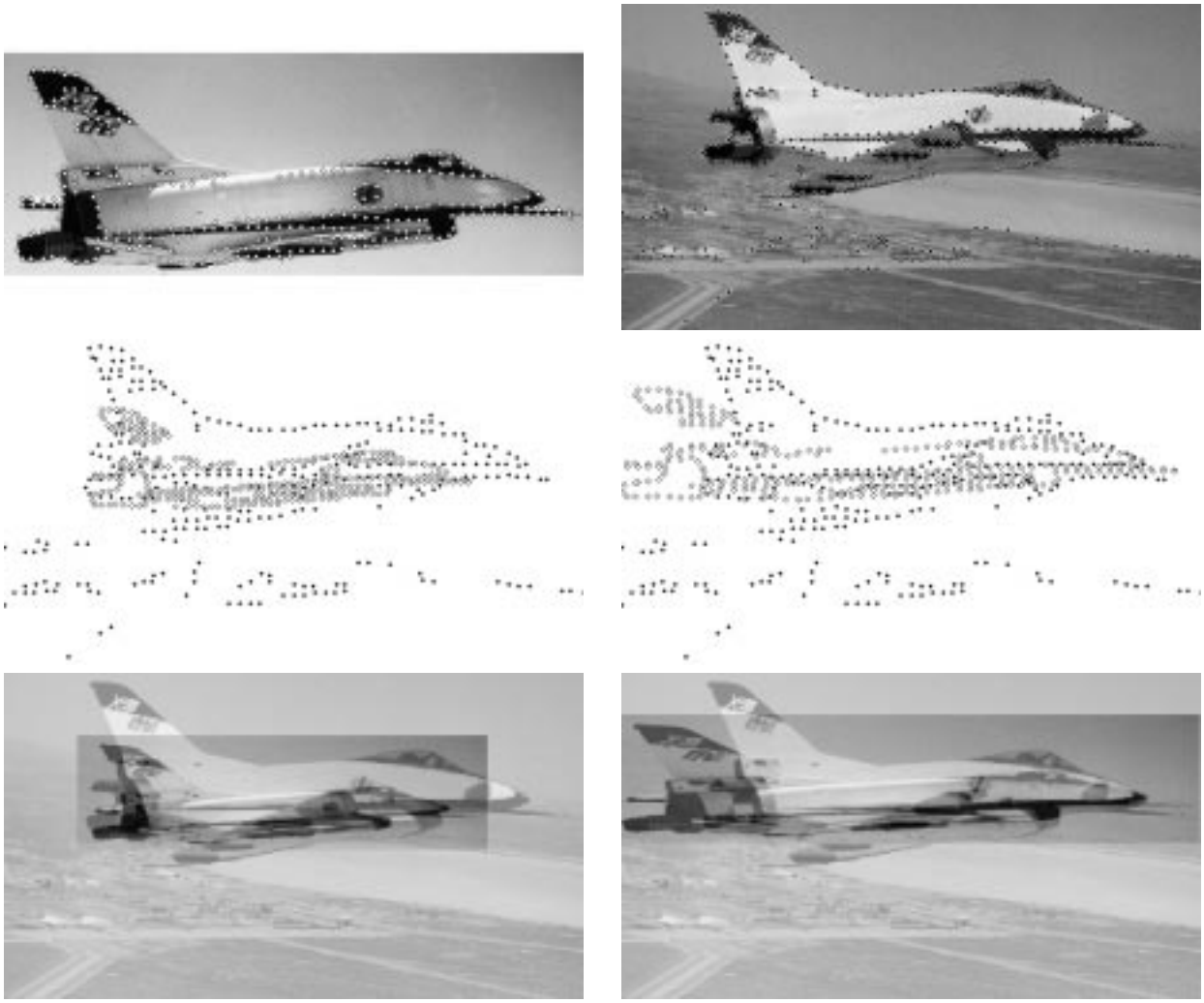

Figure 6: Different side views of the same plane.

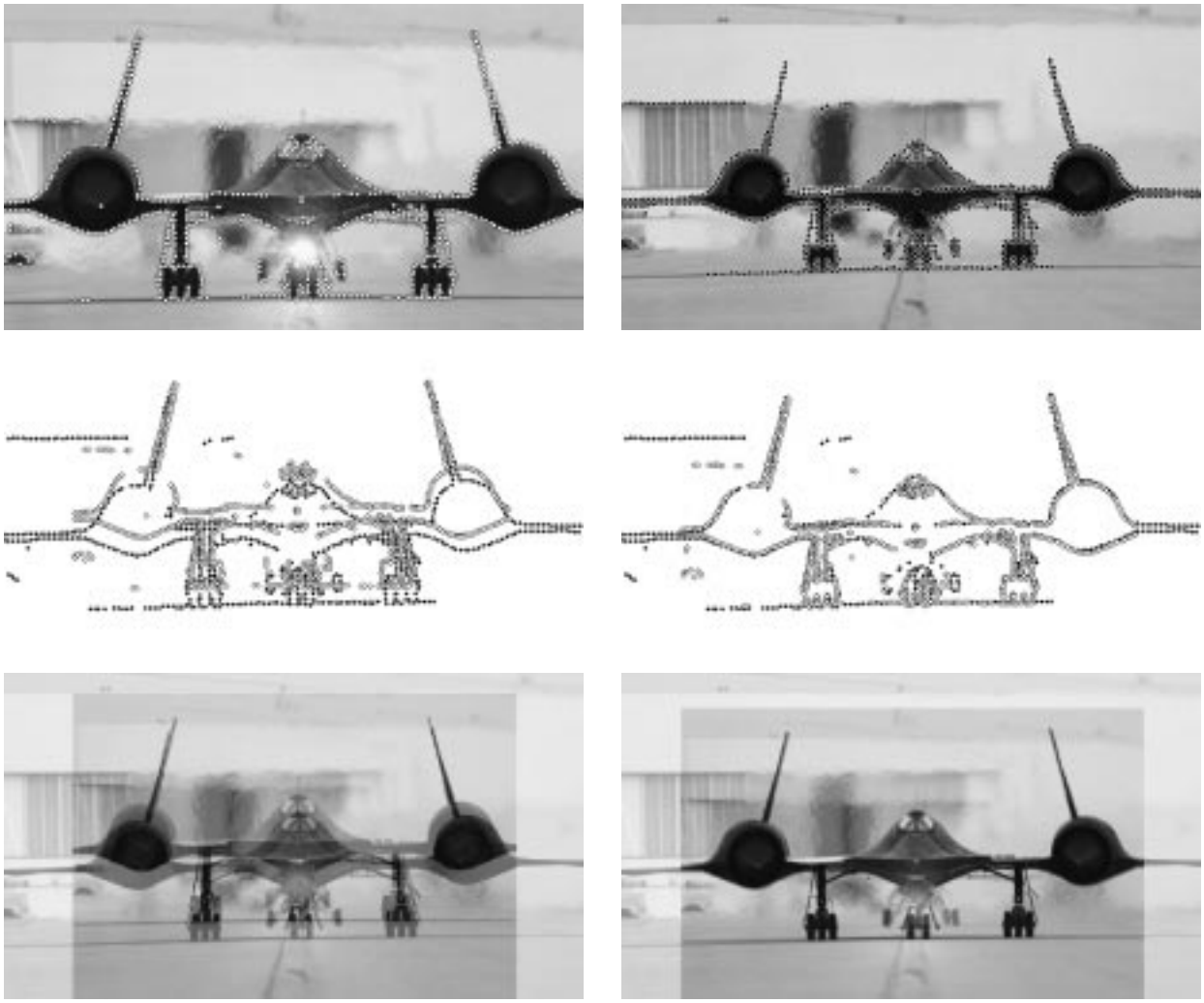

Figure 7: Two frontal views at different distances.

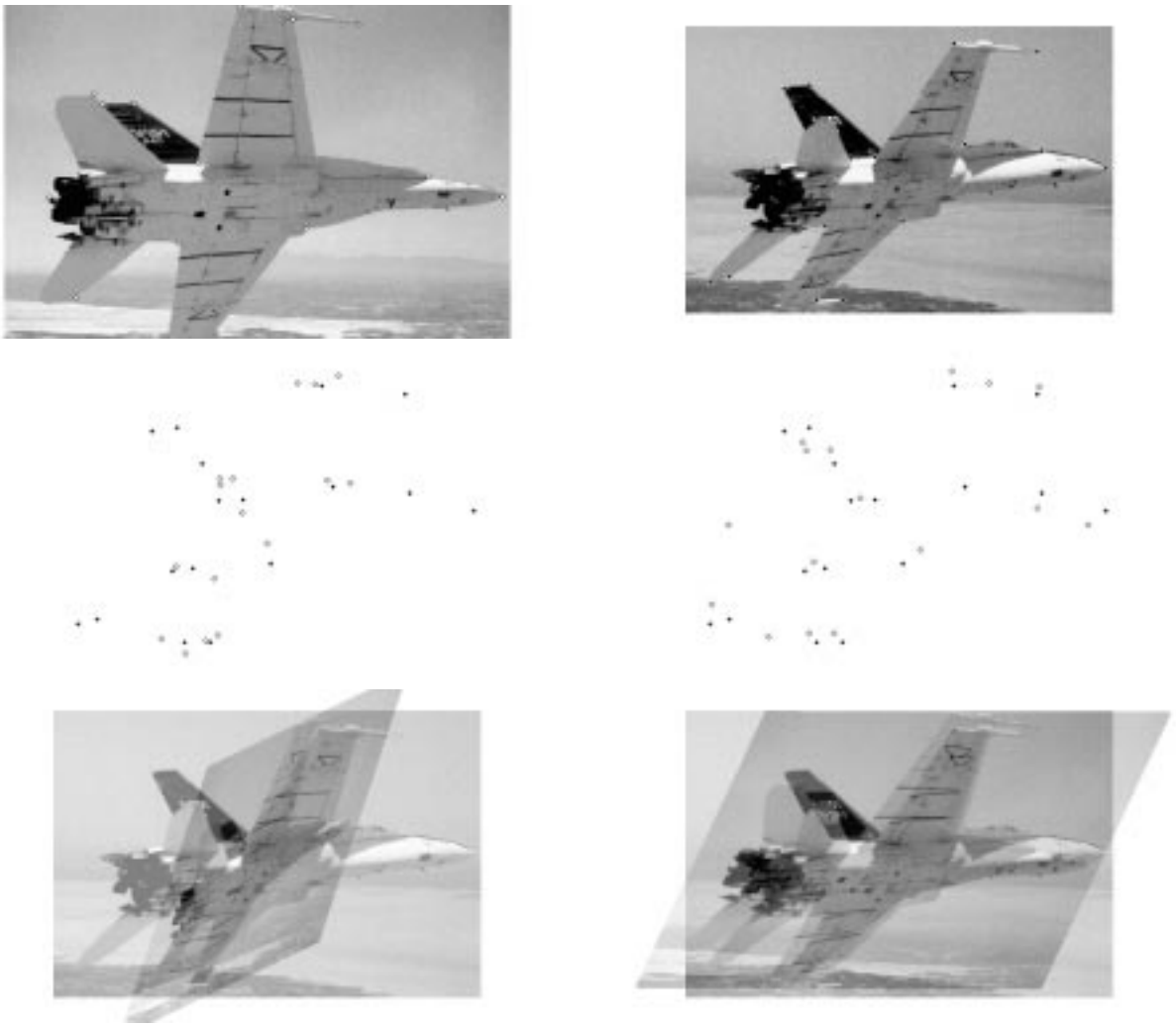

Figure 8: Two different view of the same F18 fighter.

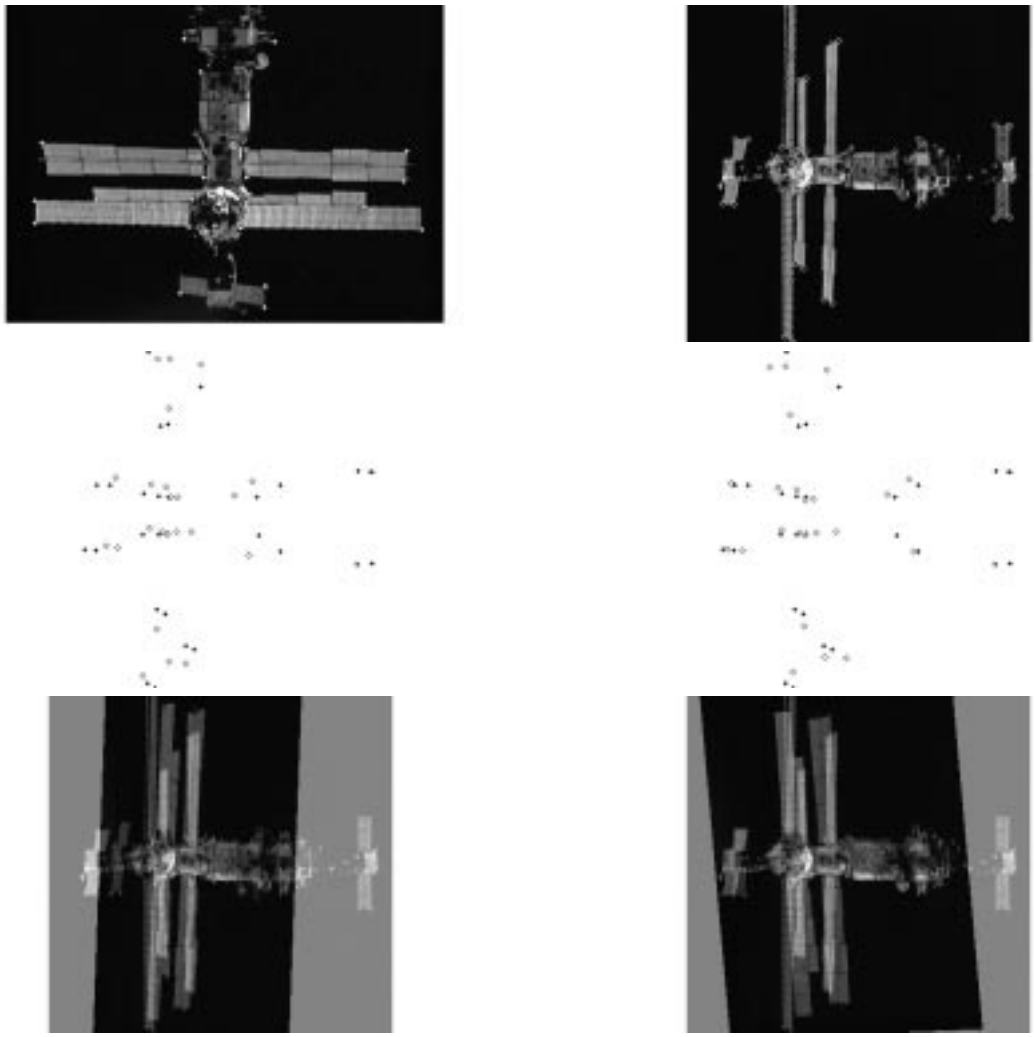

Figure 9: Two different views of the Mir station.

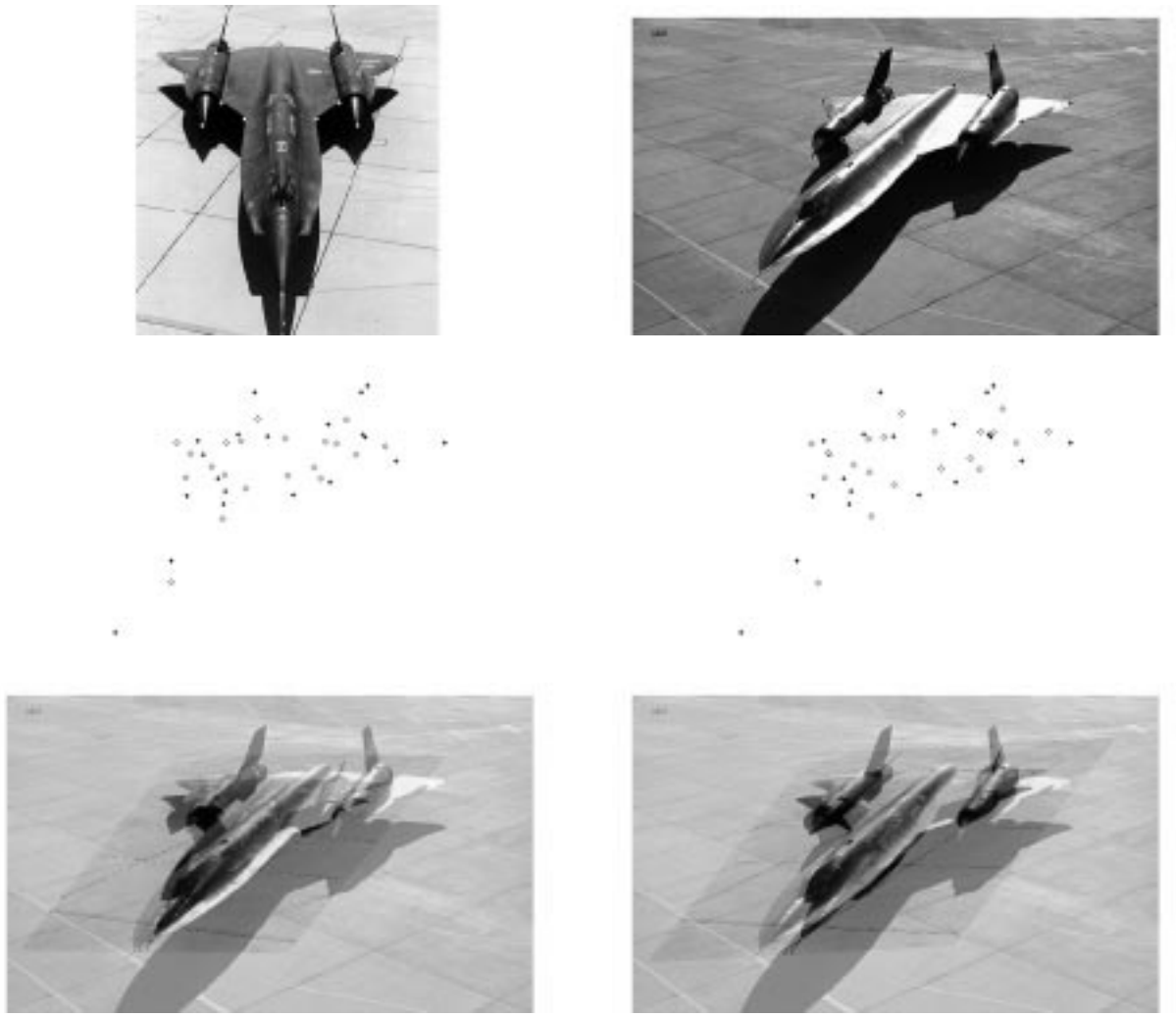

Figure 10: Two different views of the same reconnaissance plane.

5 Conclusion

In Section 2, we introduced a new similarity measure called the absolute difference, which has many advantages over other similarity measures. We prove that the absolute difference is a metric, which agrees with the intuitive notion of similarity. We also show that this measure is invariant under full affine transformations. This means the absolute difference is useful for pattern matching under affine transformations. The minimum absolute difference under full affine transformations is itself a metric which can be used to compute similarity of "affine shapes". The absolute difference is defined on patterns having multiple connected components and is robust for both perturbation and occlusion.

Section 3 discussed the general pattern matching algorithm which can be used for various similarity measures, provided particular bounds are given for them. The algorithm can be used to compute a transformation that approximates the global minimum of the similarity measure within any chosen accuracy. Our algorithm is based on a top-down subdivision of transformation space, avoiding explicit consideration of all local minima. We give bounds that allow the algorithm to be used for both the Hausdorff distance and the absolute difference. Finally, algorithms for computing bounds on the absolute difference between finite unions of hyperblocks are presented.

Section 4 gives results obtained using implementations of our algorithm for both the Hausdorff distance and the absolute difference. These results confirm that the absolute difference can be used to match reliably under all affine subgroups, even in cases where perturbation and occlusion occur. In contrast, the Hausdorff distance only performs well only under groups of isometries and limited occlusion.

In the definition of the absolute difference, we constructed functions on solid patterns, and compared these functions. There are other equivariant constructions of such functions, resulting in alternative affine invariant metrics, possibly having better properties than the absolute difference as we defined it. No exact algorithm for minimisation of the symmetric difference has been found yet, let alone for the absolute difference. These matters provide ideas for further research.

A Proof of Lemma 2

By the principle of substitution of variables in integrals, we get that the result of the action is always an integrable function. This means the class of integrable functions is closed under the proposed action.

First, we show that the identity element e of the group of C^1 diffeomorphisms acts as the identity on functions $\mathbf{a} : \mathrm{R}^d \to \mathrm{R}$:

$$\begin{aligned} e\langle \mathbf{a} \rangle(x) &= r_e(e^{-1}(x))\mathbf{a}(e^{-1}(x)) \\ &= r_e(x)\mathbf{a}(x) \\ &= \left|\det(D_e(x))\right|^{-1} \mathbf{a}(x) \\ &= \mathbf{a}(x). \end{aligned}$$

Second, we show that composition of group elements coincides with repeated application. For C^1 diffeomorphisms $g, h : \mathrm{R}^d \to \mathrm{R}^d$ the following relation between r_{gh}, r_h and r_g holds:

$$\begin{aligned} r_{gh}((gh)^{-1}x) &= \left|\det(D_{gh}((gh)^{-1}(x)))\right|^{-1} \\ &= \left|\det(D_g(h(gh)^{-1}(x))D_h((gh)^{-1}(x)))\right|^{-1} \\ &= \left|\det(D_g(g^{-1}(x))D_h(h^{-1}g^{-1}(x)))\right|^{-1} \\ &= \left|\det(D_g(g^{-1}(x)))\right|^{-1} \left|\det(D_h(h^{-1}g^{-1}(x)))\right|^{-1} \\ &= r_g(g^{-1}(x))r_h(h^{-1}g^{-1}(x)) \end{aligned}$$

Using the latter equation, we obtain:

$$\begin{aligned} g\langle h\langle \mathbf{a} \rangle \rangle(x) &= r_g(g^{-1}x)h\langle \mathbf{a} \rangle(g^{-1}(x)) \\ &= r_g(g^{-1}x)r_h(h^{-1}g^{-1}(x))\mathbf{a}(h^{-1}g^{-1}(x)) \\ &= r_{gh}((gh)^{-1}(x))\mathbf{a}((gh)^{-1}(x)) \\ &= gh\langle \mathbf{a} \rangle(x). \end{aligned}$$

B Proof of Lemma 3

Composition of the derivatives in x of two C^1 diffeomorphisms g and g^{-1} gives

$$D_g(g^{-1}(x))D_{g^{-1}}(x) = D_{gg^{-1}}(x) = D_e(x) = \mathrm{id}.$$

From this, we get the following equation:

$$D_{g^{-1}}(x) = D_g(g^{-1}(x))^{-1}. \tag{1}$$

Let $\mathbf{a} : \mathrm{R}^d \to \mathrm{R}$ be a Riemann integrable function. We need to show that $\|\mathbf{a}\| = \|g\langle \mathbf{a} \rangle\|$ for all C^1 diffeomorphisms g. Using Equation 1 and substitution

of variables in a finite-dimensional integral, we derive:

$$\begin{aligned}
\|\mathbf{a}\| &= \int_{\mathbb{R}^d} |\mathbf{a}(y)|\, dy \\
&= \int_{\mathbb{R}^d} \left|\det(D_{g^{-1}}(x))\right| \left|\mathbf{a}(g^{-1}(x))\right|\, dx \\
&= \int_{\mathbb{R}^d} \left|\det(D_g(g^{-1}(x)))\right|^{-1} \left|\mathbf{a}(g^{-1}(x))\right|\, dx \\
&= \int_{\mathbb{R}^d} r_g(g^{-1}x) \left|\mathbf{a}(g^{-1}(x))\right|\, dx \\
&= \int_{\mathbb{R}^d} \left|r_g(g^{-1}(x))\mathbf{a}(g^{-1}(x))\right|\, dx \\
&= \int_{\mathbb{R}^d} |g\langle\mathbf{a}\rangle(x)|\, dx \\
&= \|g\langle\mathbf{a}\rangle\|.
\end{aligned}$$

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. In *Proceedings of the 7th International Symposium on Algorithms and Computation*, pages 115–125, 1996.

[3] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. In *Proceedings of the 7th Annual ACM Symposium on Computational Geometry*, pages 186–193, 1991.

[4] Helmut Alt, Ulrich Fuchs, Günter Rote, and Gerald Weber. Matching convex shapes with respect to the symmetric difference. In *Algorithms ESA '96, Proceedings of the 4th Annual European Symposium on Algorithms, Barcelona, Spain, September '96*, pages 320–333. LNCS 1136, Springer, 1996.

[5] 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.

[6] D. Montgomery and L. Zippin. *Topological Transformation Groups*. Interscience tracts in pure and applied mathematics'. Interscience Publishers, Inc., 1964.

[7] 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.

[8] David M. Mount and Angela Y. Wu. On the area of overlap of translated polygons. *Computer Vision and Image Understanding*, 64:53–61, July 1996.

[9] Mark de Berg, Olivier Devillers, Marc van Kreveld, Otfried Schwarzkopf, and Monique Teillaud. Computing the maximum overlap of two convex polygons under translation. In *Proc. 7th Annu. Internat. Sympos. Algorithms Comput.*, 1996.

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

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

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

[13] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 110–120, 1992.

[14] J. Sprinzak and M. Werman. Affine point matching. *Pattern Recognition Letters*, 15:337–339, 1994.

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

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

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

[19] S. Mori, C. Suen, and K. Yamamato. Historical review of ocr research and developement. In *IEEE Proceedings*, pages 1029–1058, 1992.

[20] Christopher G. Small. *The Statistical Theory of Shapes*. Springer Series in Statistics. Springer, 1996.

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

[22] H. J. Wolfson. Model based object recognition by 'geometric hashing'. In *Proceedings of the 1st European Conference on Computer Vision*, pages 526–536, 1990.