Structural matching in computer vision using probabilistic reasoning

W.J. Christmas

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

Vision, Speech and Signal Processing Group
Department of Electronic and Electrical Engineering
University of Surrey
Guildford, Surrey GU2 5XH, U.K.

September 1995

Summary

This thesis describes a method, based on probabilistic reasoning, for matching geometric features extracted from a 2-D image with features of a similar type from some model. The feature measurements are represented in a form that is invariant (or nearly so) to some unknown transformation between scene and model spaces. Thus it may be that some measurements can be used directly; others will have to be represented in the form of relations between pairs of features in order to achieve the invariant property. The scene and model features and measurements may thus be viewed as a pair of attributed, relational graphs, and the matching problem can then be viewed as one of graph matching.

The model nodes are used as a set of candidate labels for the scene nodes (the objects). The problem is then posed as the calculation of all of the labelling probabilities for each object given the measurement information (the posterior probabilities), and the labelling that yields the highest probability is selected as the correct one. In order to make the problem tractable, by making certain independence assumptions, the conditional probabilities are expressed in terms of the measurement error distributions and some prior probabilities. The resulting formula may then be iterated, in a manner akin to a traditional probabilistic relaxation process. The form of the measurement error distributions is dependent on the type of geometric feature, the measurement noise model and the nature of the unknown scene-to-model transformation: some examples are presented.

A number of variations on the basic labelling algorithm are described, of which some have implications for real-time applications. The algorithm can also be readily implementated on several different types of parallel-processing computers.

Key words: Matching, Labelling, Probabilistic Relaxation, Object Recognition.

Email: w.christmas@ee.surrey.ac.uk

WWW: http://www.surrey.ac.uk/

Acknowledgements

I would like to thank my supervisors, Josef Kittler and Maria Petrou, for their guidance and stimulating discussions during the course of this work, and for providing the ideas and motivation that led to the work in the first place. I would also like to thank my other colleagues in the VSSP Group, for their interest and discussions. In particular thanks are due to George Matas and Paul Hoad for providing the tools and libraries that made implementing the ideas feasible, and to Andrew Stoddart for his enthusiasm for delving into the theory behind the work. And lastly and most importantly I would like to thank Shenka, my wife, for all her support and encouragement during difficult times; without that, the very notion of discovering the world of academic research would have remained but a pipe-dream.

This work was supported under the following projects:

IED and SERC joint project, no. IED-1936

EPSRC project no. GR/J89255

ESPRIT project no. 7108.

The aerial images were kindly provided by the Defence Research Agency, RSRE, UK. Thanks are also due to British Gas, who provided support for the application described in Section 9.4.

Contents

Chapter 1

An introduction to two-dimensional feature matching

Matching is a task that appears as a component of many problems in computer vision. It typically arises when we wish to find some sort of correspondence between an image and some model of some aspect of the image, or between two images. Thus it is needed for 2-D and 3-D object recognition, for depth recovery from binocular or motion stereo, for image fusion and registration, and for many other problems. In this work, we restrict the range of applications to those in which we are required to establish a correspondence between a 2-D image and a 2-D model.

In order to describe the matching problem, we start by considering the two sources of information: the *scene* and the *model*. The scene in this case is in the form of an image, which for example could contain some objects of interest, or an image taken from a vehicle of its surroundings, or a view of the ground taken from a satellite or aeroplane. The model is usually some sort of idealised representation of some characteristic *features* to be found in the scene; in particular we consider cases in which the scene and model include the spatial locations of the features. Take for example the scene in Fig. 1.1(a); this is a single-band infrared image of the ground taken from an aeroplane, stored as a grey-scale image. Since the features of interest in this case are the roads, the corresponding model is the road map shown in Fig. 1.1(b). In this example, the model covers a much larger area than the scene, and includes the whole of the (rotated) scene region. We can also stretch our definition of a model a little to include a second image; thus the scene and model could be, for example, the two images of a stereo pair, of which one (the "model" image) is regarded as a reference image.

Our task is then to find some correspondence between the scene and model. This task consists of three parts: extracting a set of features from the scene, extracting from the model a second set of features of comparable type, and matching the extracted features from the scene with those from the model. It is this last task, of matching the features, that is the subject of this work — we have assumed that some means already exists for finding the relevant scene and model features. We consider here features of a simple geometrical type, such as points, line segments or corners.

In any non-trivial application there is some unknown transformation between the scene and model spaces. We have mostly concentrated on applications in which this unknown trans-

(a) scene (b) model

Figure 1.1: Example of a scene and corresponding model

formation is a 2-D Euclidean transformation; in other words, there is (at most) an unknown translation and rotation. We also show how the method can be extended to incorporate other transformations provided that they are reasonably close to being Euclidean. Thus for example the method can handle a scaling error, provided that the error is not too large.

In the example of Fig. 1.1, we process the aerial image to extract the roads in the form of pixel strings, and then approximate these strings as a set of straight line segments, discarding those that are shorter than some threshold. These line segments constitute the scene feature set. We also create a second set of line segments that are an approximation of the roads from the map; these form the model feature set. These two sets are shown in fig 1.2. We then look for corresponding pairs of features in the two sets. In this example, this set of corresponding pairs could then be used, if desired, to locate the aerial image on the map.

In this work, the approach we have taken to find the correspondences is based on probabilistic reasoning. The model features are used as a set of potential labels for the scene features. We compute the probabilities of the various possible labellings, taking into account all of the measurement information available to us. We then take the view that the best labelling is the one with the highest probability (the Maximum A Posteriori probability). In order to calculate the probabilities, a Bayesian approach is used to reformulate the probabilities in terms of the measurement distributions: these distributions provide a direct means of incorporating the measurements in to the calculation. Much reliance is placed on the use of relations between pairs of features, in order to capture the contextual information in both scene and model. The methodology also handles many-to-one labellings, which are frequently required in a computer vision context. This approach leads to an evidence-combining formula which prescribes in a unified and consistent manner how measurements relating to both individual features and pairs of features, together with any available prior world knowledge, should be jointly brought to bear on the labelling problem.

(a) scene (b) model

Figure 1.2: Scene and corresponding model with extracted line segment features

The resulting update rule can then be iterated to ensure a consistent solution. The iterated rule is seen to have many parallels with more traditional probabilistic relaxation approaches to feature matching, although our method appears to converge more quickly. The derivation of our rule from a simple probabilistic statement removed many of the heuristic elements of the traditional relaxation methods; in particular we provide a rationale for the calculation of the compatibility coefficients. It also offers a clear methodology for designing such processes.

The compatibility coefficients that characterise probabilistic relaxation algorithms are defined in this case in terms of the relation measurement error distributions. Hence measurements are incorporated into all iterations of the relaxation process via these compatibility coefficients — this represents the most significant point of departure from the earlier work of [40]. The support function, which is also derived from the formulation, rather than being *ad hoc*, is in the form of a product rule; we show how this form can be approximated by the summation rule (as used by Rosenfeld *et al.* [50] and others) in cases where the contextual information is low. The theoretical framework also shows how the prior information should be used to initialise the probabilistic relaxation process. This contrasts with the approach adopted by Li [42] who commenced the iterative process from a random assignment of label probabilities.

The feature matching method that we present here owes its early inspiration to the probabilistic relaxation scheme of Kittler and Hancock [40]; however the introduction in this work of relations between the features makes the method applicable to a much wider range of matching applications. The use of relations follows in particular from the work of [42, 43]. The algorithm evolved via [37, 38] into its present form [13, 15]. The non-iterative version was described in [18], with a more complete description in preparation [12]. Enhancements to the algorithm appeared in [16, 17, 19], and an application was described in [48]. The rela-

tionship of the method to neural network approaches appeared in [14, 36].

In what follows, we begin by setting out the framework for the matching theory in Chapter 2. We then develop the theory for the labelling scheme in Chapter 3, and discuss the effects of relaxing the scheme in Chapter 4. The evaluation of the terms in the labelling scheme is covered in a general way in Chapter 5, and is then described in detail for specific feature types in Chapter 6. Chapters 7 and 8 deal with various modifications that can be made to the algorithm, those in Chapter 7 being concerned with real-time issues. In Chapter 9 the algorithm is tested on a selection of different applications, and sensitivity to parameter values is examined. Some implementation issues are briefly covered in Chapter 10. We deferred the discussion of other matching algorithms and their relationship with the one described here to Chapter 11, on the grounds that it is easier to make the comparisons once our method has been fully described. Finally we make some observations on the method in Chapter 12, and discuss some ways in which the work could be continued.

Chapter 2

Object labelling and graph matching

In this chapter we first express the matching problem as one of *object labelling*; then, by considering the type of measurement information available to us, we show how it is also a problem of matching attributed relational graphs. We first discuss the feature sets, and define what we mean by labelling in this context. We then discuss in a general sense the measurements that can be made on the features in the scene and model and the transformation between the scene and model measurement spaces. Uncertainties in this transformation lead to the idea of relations between pairs of measurements, giving rise to the idea that the scene and model measurement information can be represented as a pair of attributed relational graphs. Finally we give examples showing how different types of attributes and relations are appropriate to different levels of knowledge about the transformation between scene and model measurement domains.

2.1 Feature sets and object labelling

Consider a scene from which we have by some means extracted a set of features. We refer to a scene feature as an *object*,[1] and we call the set of scene features the object set, O, of N members, *i.e.*

$$O = \{O_1, O_2, \ldots, O_N\}$$

Similarly there is a set of model features, which we use as *labels* for the objects. Ideally we would like to label all of the objects in a single operation, in order to ensure that the labellings are consistent with each other; this is known as the "message-centred" approach. In practice though we are unable to find an algorithm that can achieve this reliably in a reasonable time. We therefore turn our attention instead to a method that labels each object individually — the "object-centred" approach. (Later on in Chapter 4 we look at a method of ensuring consistency between the individual labellings.)

We define a distinct label set Ω_i for each object O_i in O. These N labels sets may, but need not, consist of the same members. Either way, we need to distinguish between the different sets

[1] The term "object" is frequently used to denote some more complex structure that corresponds to some complete physical entity; here it refers to a single feature.

because in the derivation of the labelling rule (Chapter 3) we need to be able to distinguish labels of different objects. On the other hand, where the distinction between the label sets is not important, the suffix is dropped.

We use the symbols $\alpha_i, \beta_i, \ldots$ to denote specific labels from the set Ω_i, although the object suffices of the labels are frequently omitted to avoid clutter. The symbol $\mathcal{L}_i^{\alpha}$ denotes the labelling of object O_i with label α_i. The symbols ω_i or υ_i denote a general label, or label "variable", from the set Ω_i, typically used when we perform some operation over the whole label set Ω_i.

In each label set Ω_i there are $M_i + 1$ features, M_i of which are obtained directly from the model; the remaining member is the *null* label $\emptyset$, which is used to label objects for which no other label is appropriate. As we shall see later, the null label is unique in that it does not correspond to any physical feature extracted from the model. It has three functions; it can be used to:

- represent model features that are missing as a result of errors in the model;
- represent model features which are "outside" the model in applications where the scene covers a larger area than the model;
- act as a label for spurious scene features that are generated as a result of noisy scene images or imperfections in the feature extraction.

The consequence of including the null label is that it is then *always* possible to find a label for all of the objects, thereby making it simpler to express the labelling process in a relatively homogeneous manner.

Thus we have presented the matching process as one of labelling. We consider that each object ultimately has only one "correct" label; on the other hand, because there is a separate label set for each object, and because the same label can appear in more than one label set, the same label can be assigned to more than one object.

2.2 Feature measurements

Each scene feature O_i is characterised by a set of K_m values that are referred to here as the scene feature *measurements* $\mathbf{x}_i$:

$$\mathbf{x}_i = \left\{ x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(K_m)} \right\}$$

We also use the unsubscripted variable $\mathbf{x}$ to denote the complete set of measurement sets:

$$\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$$

These measurements will usually include the position of the feature within the image, and (depending on the application) may include others, *e.g.* the orientation, size or colour of the feature. Because the scene image will usually contain noise, and because the feature extraction process is subject to errors, the measurements themselves will also contain errors, and

can therefore be viewed as instances of some corresponding random variables. We might note however that these random variables are not necessarily independent, and it may be necessary to specify a joint distribution for them. As is customary, we frequently use the same symbol to denote both a measurement and the random variable of which it is an instance.

We can also make a corresponding set of K_m measurements $\mathbf{X}_\alpha$ on each model feature α:

$$\mathbf{X}_\alpha = \left\{ X_\alpha^{(1)}, X_\alpha^{(2)}, \ldots, X_\alpha^{(K_m)} \right\}$$

and similarly we abbreviate the set of all label measurement sets thus:

$$\mathbf{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_M\}$$

In keeping with the usual notion of a model as some sort of reference, we adopt the convention that the model measurements do not contain errors, and so do not have corresponding random variables. In applications for which this is not the case, the model errors are assumed to have been incorporated into the scene measurements.

Further discussion of the measurements has to be postponed until we have considered what we know of the relationship between the scene and model measurement spaces.

2.3 The relationship between the scene and model measurements

In order to perform the labelling, we have to compare scene and model measurements. To do this, the measurements must be expressed in the same domain, for example by transforming the scene measurements to the model domain (or vice versa). This transformation usually has two components, known and uncertain. Thus in the example in Chapter 1 we may know the relative scale of the scene and model, but the relative displacement may only be known approximately — the image lies somewhere within the map — and the relative orientation not at all. We assume that all of the available certain information about the transformation has already been applied (*e.g.* the image has been scaled to match the map). We are then left with the uncertain component of the transformation, which forms the crux of the labelling problem.

We denote by Φ the uncertain transformation operator that maps measurements from the scene space to the model space. Thus if we have some set of measurements $\mathbf{x}_i$ associated with object O_i, then these measurements can be expressed in the model domain by a set $\mathbf{x}_i'$, where

$$\mathbf{x}_i' = \Phi \mathbf{x}_i$$

In other words Φ represents the knowledge we have that enables us to relate measurements in the scene to those in the model. In general we know something about Φ but cannot define it completely – it may for example include parameters whose values are unknown, or parameters that may be represented as random variables.

There are two extremes, in which either Φ is defined precisely or nothing is known about it at all. If Φ is completely unknown, we cannot solve the matching problem at all, since we have no means of comparing measurements in the two spaces. On the other hand, when

we know Φ precisely, the problem becomes trivial because scene measurements are then mapped onto the model domain precisely, and the labelling process reduces to one of looking for the "nearest" label to each object.

Most of the time, however, our knowledge of Φ is uncertain but nevertheless significant, and usually we can explicitly represent the uncertainty in Φ in the form of a finite set of K_Φ uncertain (but constant) parameters, $\{c^{(1)}, \ldots, c^{(K_\Phi)}\}$. These parameters might for example represent the relative position or orientation of the two sets of features. We distinguish between two categories of these parameters. We may know nothing at all about a parameter $c^{(l)}$. Alternatively for some other parameter $c^{(m)}$ we will know *something* about its value. In the former case, we cannot use the measurements directly; we must resort to using some combination of them. In the latter case, we may know for example that $c^{(m)}$ lies within some finite range; we can incorporate this information by modelling $c^{(m)}$ as a random variable, perhaps using maximum entropy principles to estimate a possible distribution for it.

Consider for example a simple application in which the features are points, and in which there are two measurement types ($K_m = 2$): the components of the position vector of the features. In this case, the operator Φ reduces to the addition of a constant vector to the scene measurements to map them onto the model space. If the location of the scene with respect to the model is uncertain, this vector is uncertain, and so we have two uncertain transformation parameters ($K_\Phi = 2$). If nothing at all is known about the parameter values, they are of no direct use towards the labelling process at all; if something is known about them, they can be used to provide some information towards the labelling process. On the other hand, there may be some *differential* information that we can derive from the measurements from two or more objects that is invariant to the uncertain transformation, and can be usefully compared with corresponding quantities derived from the model. It is the use of this differential information, and the way in which we incorporate it into the labelling process, that represents the main advance over previous MAP labelling methods.

2.4 Higher-order measurements; attributed relational graphs

In the above discussion, we concluded that in general we cannot necessarily use all of the measurement types directly to solve the matching problem; this is because the uncertainty in the transformation Φ means that we cannot in general map all of the measurements from the scene domain to the model domain without loss of information. What we can do instead is to combine measurements from ensembles of objects, to generate quantities known as *relations* that are invariant to the transformation Φ. A relation between n objects is called an n-ary relation. Although in principle any order of relation may be used for our purposes, there is a rapidly-increasing computational cost as n increases; we therefore only consider applications that can be solved using binary relations alone (*i.e.* $n = 2$). Examples of binary relations are: the relative position of one object with respect to another, or relative size or orientation. Topological or symbolic relations (*e.g.* "object O_i is on the top of object O_j") are not in general considered here. (However when we consider methods of improving the performance of the matching algorithm in Chapter 7, we do in effect consider relations of the type "object O_i is within range of object O_j". It will be evident from this that symbolic relations could easily be incorporated if desired.) We note that, because we are considering

binary relations only, it is not always possible to find sufficient relations that are invariant to the transformation; in such cases, the loss of information is likely to be severe.

We may regard the set of objects as a set of nodes in a graph (the scene graph), for which we can define sets of attributes and relations. Consider first those object measurement types ($K_{\mathbf{a}}$ of them, where $K_{\mathbf{a}} \leq K_m$) that can be used directly; that is, there is sufficient information in Φ that the object measurements can be usefully compared with the model measurements directly. Then the set of such measurements that are associated with each object O_i are defined as the *attributes* $\mathbf{a}_i$ of the graph node O_i:

$$\mathbf{a}_i = \{a_i^{(1)}, \ldots, a_i^{(K_{\mathbf{a}})}\}$$

In other words there are $K_{\mathbf{a}}$ attribute types associated with the particular matching problem. We denote by $\mathbf{a}$ the entire set of scene attributes:

$$\mathbf{a} = \{\mathbf{a}_1, \ldots, \mathbf{a}_N\}$$

In some applications it may be more convenient to select $K_{\mathbf{a}}$ attributes that are some function of the relevant $K_{\mathbf{a}}$ measurements, rather than to use the measurements as attributes directly.

Similarly, the binary relations derived from pairs of object measurements may be regarded (also after mapping into the model space) as the relations on the arcs joining the nodes. There are $K_{\mathbf{r}}$ relations $r_{ij}^{(k)}$ of different types between each pair of nodes O_i and O_j:

$$\mathbf{r}_{ij} = \{r_{ij}^{(1)}, \ldots, r_{ij}^{(K_{\mathbf{r}})}\}$$

where (provided that measurement information is not duplicated within the relations) we would expect that $K_{\mathbf{r}} \leq K_m$. If all of the measurement information is expressed in the attributes and relations, we would expect that the total number of attribute and relation types will not be less than the number of measurement types, *i.e.*

$$K_{\mathbf{a}} + K_{\mathbf{r}} \geq K_m$$

The set of all relations between node O_i and all of the other nodes is denoted by $\mathbf{r}_i$, where

$$\mathbf{r}_i = \{\mathbf{r}_{i1}, \mathbf{r}_{i2}, \ldots, \mathbf{r}_{ii-1}, \mathbf{r}_{ii+1}, \ldots, \mathbf{r}_{iN}\}$$

and the entire set of object relations, between all possible pairs of nodes, is denoted by $\mathbf{r}$. Thus the scene graph is an attributed, relational graph.

A corresponding model graph can also be constructed from the model labels and measurements, where model node α has $K_{\mathbf{a}}$ attributes $\mathbf{A}_\alpha$:

$$\mathbf{A}_\alpha = \{A_\alpha^{(1)}, \ldots, A_\alpha^{(K_{\mathbf{a}})}\}$$

and there are $K_{\mathbf{r}}$ relations $\mathbf{R}_{\alpha\beta}$ between nodes α and β, where

$$\mathbf{R}_{\alpha\beta} = \{\mathbf{R}_{\alpha\beta}^{(1)}, \ldots, \mathbf{R}_{\alpha\beta}^{(K_{\mathbf{r}})}\}$$

We can see therefore that the labelling problem can also be seen as one of graph matching. If relations are computed between all possible pairs of nodes, the graph is fully connected.

As we have seen, the attributes and relations are derived from noisy measurements, which have been mapped into the model space with some additional uncertainty due to the uncertain transformation Φ; thus the attribute set $\mathbf{a}_i$ can be viewed as instances of some underlying random variables $[\mathbf{a}_i]$, whose distribution is made up of two components, due to two entirely different causes:

- measurement noise, which is generally uncorrelated between objects, and
- the transformation between scene and model, whose uncertainty (when present) is generally highly correlated between objects,

and similarly for the relations. On the other hand, because of the assumption earlier that the model measurements are precise, the model attributes $\mathbf{A}$ and relations $\mathbf{R}$ can be calculated precisely from the model measurements. Now we would expect that, for each object O_i, there should be some label α_i from the set Ω_i that is its correct label; for this label the attributes $\mathbf{A}_\alpha$ should then correspond to the expected value of $[\mathbf{a}_i]$. The same argument can be made for the relations.

Because the relations (and possibly the attributes) are created from functions of the measurements, the form of the transformation operator is different in each case; we therefore denote the transformation operators for the attributes and relations by $\Phi_\mathbf{a}$ and $\Phi_\mathbf{r}$ respectively.

In order to avoid further confusion, we should note that the terms "scene feature", "object" and "scene node" all refer to the same thing; the different terms merely provide a difference of emphasis. The terms "model feature", "label" and "model node" are likewise synonymous.

2.5 Some examples

Consider the hypothetical application discussed above (Section 2.3), in which the nodes represent point features, and in which each object O_i has two associated (positional) measurements $\mathbf{x}_i = (x_i, y_i)$; that is, $K_m = 2$.

Example 1 *transformation known precisely ($K_\Phi = 0$).*

In this case, as was stated earlier, the problem is relatively trivial. Since $\Phi_\mathbf{a}$ is known precisely (*i.e.* its distribution is a delta function), we use the transformed position measurements $\mathbf{x}$ as the attributes $\mathbf{a}$:

$$\mathbf{a}_i = \mathbf{x}_i$$

and no relations are needed ($K_\mathbf{a} = 2$, $K_\mathbf{r} = 0$). Because of the measurement noise, the correct labels will not correspond exactly; instead, to find the best labelling, we look for labels that minimise the Mahanalobis distance between the scene and model attributes.

Example 2 *translation unknown ($K_\Phi = 2$).*

We can no longer use the measurements as attributes, because we do not know how to map the scene measurements into the model space. However we can use relations in the form of the differences between pairs of measurements, since these will be invariant to the unknown translation:

$$\mathbf{r}_{ij} = \left(\mathbf{x}_j - \mathbf{x}_i\right)$$

Thus $K_{\mathbf{a}} = 0$, $K_{\mathbf{r}} = 2$.

Example 3 *translation uncertain ($K_\Phi = 2$).*

We know the translation between scene and model spaces, but only approximately (for example we may know that the translation lies between some limits). We therefore use the transformed position measurements as attributes, but we also use the difference relations of the previous example, since they represent the measurement information in a more accurate form ($K_{\mathbf{a}} = 2$, $K_{\mathbf{r}} = 2$). In this example we say that the attributes are *weak* and the relations *strong, i.e.* the relations have a relatively narrow distribution compared with the attributes.

Example 4 *translation and orientation unknown ($K_\Phi = 3$).*

In this case, the scene is subjected to an unknown Euclidean transformation with respect to the model. We can now only use the scalar difference between the position measurements as a relation ($K_{\mathbf{a}} = 0$, $K_{\mathbf{r}} = 1$):

$$\mathbf{r}_{ij} = \left|\mathbf{x}_j - \mathbf{x}_i\right|$$

The information pertaining to relative orientation, which is half the total available measurement information, is no longer of any use; higher-order relations would be needed to express this lost information.

Example 5 *translation, orientation and scale unknown ($K_\Phi = 4$).*

In other words, this corresponds to an unknown similarity transformation. There are no second-order relations that are invariant to this transformation, so none of the measurement information is of any use; we therefore cannot solve this problem without recourse to higher-order relations ($K_{\mathbf{a}} = 0$, $K_{\mathbf{r}} = 0$).

Chapter 3

Theoretical framework for object labelling using Bayesian reasoning

In this chapter, by formulating the labelling problem in the framework of Bayesian probability theory, we derive a formula which enables us to calculate the labelling from the given measurements. We start by expressing the labelling problem in terms of a Maximum A Posteriori (MAP) labelling probability. Then by using Bayes's rule and by making a series of assumptions, we are able to expand and factorise the MAP probability so as to express it in terms of the measurement information (*i.e.* attributes and relations) available to us. The explicit use of relations as evidence in computing the contextual MAP probability of the labelling represents a crucial point of departure from previous work on probabilistic labelling [40] which relied on attributes only.

3.1 Formulation of the labelling problem as a Maximum A Posteriori labelling rule

As was mentioned in the previous chapter (Section 2.1), in order to make the labelling problem tractable, we are adopting the object-centred approach; that is the labelling for each object is calculated independently from the others. We define the "best" label for an object to be the one that generates the highest MAP probability; that is, the object O_i will be assigned the label ω_i^{MAP}, provided that it is the most probable labelling given all the information we have for the system (*i.e.* all of the object and label attributes and relations). We can therefore state the MAP rule as follows: the most appropriate label of object O_i is ω_i^{MAP} where

$$\omega_i^{MAP} = \arg\left(\max_{\omega_i \in \Omega_i} \Pr\left(\mathcal{L}_i^{\omega_i} \mid \mathbf{a}, \mathbf{r}, \mathbf{A}, \mathbf{R}\right)\right)$$

In our approach, the model attributes and relations, **A** and **R**, are regarded as known constants rather than random variables, and therefore always appear as conditional quantities. For conciseness they are generally omitted in the expressions that follow, and are assumed to be implicitly included. In other words, the MAP rule is written as:

$$\omega_i^{MAP} = \arg\left(\max_{\omega_i \in \Omega_i} \Pr\left(\mathcal{L}_i^{\omega_i} \mid \mathbf{a}, \mathbf{r}\right)\right) \tag{3.1}$$

In this expression, we note that much of the measurement information may be in some sense duplicated; for example there are N^2 relations derived from only N objects.

3.2 Reformulation of the MAP rule in terms of known quantities

We do not know how to evaluate the conditional probability on the right-hand side of (3.1) directly, so we seek to express it in terms of known quantities. Because we are going to adopt an approach based on Bayes's Theorem, we would expect to incorporate the measurement information by deriving an expression for a specific conditional labelling $\Pr(\mathcal{L}_i^{\alpha}|\mathbf{a},\mathbf{r})$ based on the probability density functions (p.d.f.s) of the attributes and relations conditional on the appropriate labellings, and on some view of the prior labelling probabilities. That is, we would expect to find in the expression terms such as:

- $p(\mathbf{a}_i|\mathcal{L}_i^{\alpha})$, the p.d.f. for the attribute vector given the labelling $\mathcal{L}_i^{\alpha}$, evaluated at the point $\mathbf{a}_i$. The shape of the p.d.f. will be determined by the uncertain processes involved in generating $\mathbf{a}_i$ (*i.e.* in feature extraction and mapping into the model domain). The p.d.f. will be offset by the attribute vector $\mathbf{A}_{\omega_i}$ of the label in the match $\mathcal{L}_i^{\alpha}$.
- $p(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta})$, the p.d.f. for the relation vector given the labellings $\mathcal{L}_i^{\alpha}$ and $\mathcal{L}_j^{\beta}$, evaluated at the point $\mathbf{r}_{ij}$; its form will be determined by similar reasoning.
- $\Pr(\mathcal{L}_i^{\alpha})$, the a priori probability of the labelling $\mathcal{L}_i^{\alpha}$, *i.e.* the probability based on some reasoning that does not incorporate any knowledge of the attributes or relations.

We next consider if we can reduce the number of attributes and relations that the labelling is dependent on, by:

Shrinking the set of relations. Since the relation set $\mathbf{r}_i$ includes relations between object O_i and *all* other objects, measurements from *all* objects are used in generating $\mathbf{r}_i$. From perhaps a rather simplistic viewpoint, we might then argue that in that case adding further relations cannot provide any more information. This may not strictly be true; on the other hand because in practice we usually iterate the MAP expression (as we discuss later in Chapter 4), information from the remaining relations will be incorporated in successive iterations. We therefore reduce the number of relations that are considered for the labelling of object O_i to the set $\mathbf{r}_i$. Restricting the number of relations in this way is also consistent with the decision to pursue the object-centred approach: we are restricting the information to that which concerns the object being labelled.

Shrinking the set of attributes. We can consider two classes of attributes: those whose corresponding measurements were also used to generate the relations, denoted $\mathbf{a}'_j$, and the remaining ones, denoted $\mathbf{a}''_j$. The attributes of the former class can be computed as some function f of the relevant attributes of object O_i and the corresponding relations, *i.e.*

$$\mathbf{a}'_j = f\left(\mathbf{a}'_i, \mathbf{r}_{ij}\right)$$

and so it seems reasonable to exclude the attributes $\{\mathbf{a}'_j, \forall j \neq i\}$. In the latter class, because there are no relevant relations, the information contained in such attributes

$\mathbf{a}_j''$ of object O_j may still be of use in the labelling of object O_i; all such attributes are therefore retained.

We therefore make the assumption:

Assumption 3.1. *The labelling of object O_i is only significantly affected by the values of attributes and relations involving that object, and by those attributes of other objects for which there are no corresponding relations.*

Comment: In some applications (*e.g.* Example 2 in Section 2.5) we can in fact calculate all of the discarded attributes and relations from the remaining ones, in which case the assumption is clearly valid. In other applications the assumption may not be strictly true; however we argue that the loss of information is likely to be small, and therefore that the assumption is still a reasonable one. We also note that, as a result of this assumption, none of the measurement information is now duplicated.

Thus the conditional items in the probability term of the MAP rule (3.1) can be reduced in number:

$$\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}, \mathbf{r}\right) = \Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right) \tag{3.2}$$

where $\mathbf{a}''$ is defined as the whole set of such attributes:

$$\mathbf{a}'' = \left\{\mathbf{a}_1'', \ldots, \mathbf{a}_N''\right\}$$

Using the definition of conditional probability, we can write:

$$\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right) = \frac{p\left(\mathcal{L}_i^{\alpha}, \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right)}{p\left(\mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right)} \tag{3.3}$$

As was said above, we hope ultimately to generate an expression that includes terms for the p.d.f.s of the relations. Now the p.d.f. $p(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})$ for the relations between objects O_i and O_j includes the labelling $\mathcal{L}_j^{\beta}$ as well as $\mathcal{L}_i^{\alpha}$, whereas the terms in (3.3) only include at most a reference to the labelling $\mathcal{L}_i^{\alpha}$. We therefore use the theorem of total probability to expand both numerator and denominator over the label set for each of the possible object labellings in turn, to ensure that all labellings appear for all other objects (including the labelling $\mathcal{L}_j^{\beta}$):

$$\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right) = \frac{\displaystyle\sum_{\omega_1 \in \Omega_1} \cdots \sum_{\omega_{i-1} \in \Omega_{i-1}} \sum_{\omega_{i+1} \in \Omega_{i+1}} \cdots \sum_{\omega_N \in \Omega_N} p\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right)}{\displaystyle\sum_{\omega_1 \in \Omega_1} \cdots \sum_{\omega_i \in \Omega_i} \cdots \sum_{\omega_N \in \Omega_N} p\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_i^{\omega_i}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right)} \tag{3.4}$$

The next step is to factorise the term that appears in the numerator and denominator of the above equation. This can be done in different ways, each of which may need a different set

of further assumptions. In the approach that we favour,[1] it is factorised as follows:

$$p\left(\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N},\mathbf{a}_i',\mathbf{a}'',\mathbf{r}_i\right) = \\ p\left(\mathbf{r}_i \mid \mathbf{a}_i',\mathbf{a}'',\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) \times p\left(\mathbf{a}_i',\mathbf{a}'' \mid \mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) \times \Pr\left(\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) \quad (3.5)$$

We evaluate each of the terms in (3.5) in turn. In considering the first term, we make the assumption:

Assumption 3.2. *The relations in the set* $\mathbf{r}_i$ *are conditionally independent of each other. That is,* $\mathbf{r}_{ij}$ *by itself provides no information about* $\mathbf{r}_{ik}$.

Comment: Since relations are derived from feature measurements, all of the relations $\mathbf{r}_i$ are derived from measurements that include those made on object O_i. Thus any error on these measurements will be common to all of the relations $\mathbf{r}_i$, making it unlikely that the relations will be independent from each other. On the other hand, this common measurement information represents only half of the information used to derive the relations; the other half comes from the remaining nodes $O_{j,j\in N_i}$, whose errors are likely to be independent.

If we make the assumption, this term can be further factorised:

$$p\left(\mathbf{r}_i \mid \mathbf{a}_i',\mathbf{a}'',\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) = \prod_{\forall j\neq i} p\left(\mathbf{r}_{ij} \mid \mathbf{a}_i',\mathbf{a}'',\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) \quad (3.6)$$

We again seek to reduce the amount of conditional information. With this in mind, we make the following assumptions:

Assumption 3.3. *The relations* $\mathbf{r}_{ij}$ *do not depend on the attributes* $\mathbf{a}_i'$ *alone (*i.e. *without knowledge of* $\mathbf{a}_j'$*).*

Comment: The errors in these attributes are likely in practice to be dominated by systematic errors in transforming them to the model domain. The relations on the other hand are generally selected to be invariant to such errors, so are likely to be dominated by noise due to feature extraction. Thus the assumption is reasonable.

Assumption 3.4. *The relations* $\mathbf{r}_{ij}$ *do not depend on the attributes* $\mathbf{a}''$.

Comment: This is reasonable since the attributes $\mathbf{a}''$ were the ones that were not used in the derivation of the relations.

Using these two assumptions, (3.6) can be simplified to:

$$p\left(\mathbf{r}_i \mid \mathbf{a}_i',\mathbf{a}'',\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_N^{\omega_N}\right) = \prod_{\forall j\neq i} p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\omega_i},\mathcal{L}_j^{\omega_j}\right) \quad (3.7)$$

We next address the second factor on the right hand side of (3.5):

[1] See Section8.3 for an alternative derivation.

Assumption 3.5. *The attributes $\mathbf{a}'_i, \mathbf{a}''$ are conditionally independent of each other.*

Comment: As we have remarked previously, there could be systematic errors in the attributes which would invalidate this assumption. However it has to be made; the consequence is to ignore the systematic nature of the errors.

Then the second factor on the right hand side of (3.5) can be simplified:

$$p\left(\mathbf{a}'_i, \mathbf{a}'' \mid \mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_N^{\omega_N}\right) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\omega_i}\right) \prod_{\forall j \neq i} p\left(\mathbf{a}''_j \mid \mathcal{L}_i^{\omega_i}\right) \tag{3.8}$$

Finally we consider the last factor of (3.5):

Assumption 3.6. *The prior (unconditional) labelling probabilities are independent.*

Comment: We argue that, *without any measurement information*, knowing the label for one object tells us nothing about that of another. This assumption is made more plausible by the decision to allow the same label to be used for more than one object (Section 2.1), a consequence of our object-centred approach. One might think that, if a given object has a particular label, other objects are less likely to have that label. On the other hand, we do encounter situations in practice in which it is important that more than one object can have the same label, and this assumption reflects such situations.

We can then factorise the third factor of (3.5):

$$\Pr\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_N^{\omega_N}\right) = \prod_{\forall j} \Pr\left(\mathcal{L}_j^{\omega_j}\right) \tag{3.9}$$

So, substituting (3.7), (3.8) and (3.9) into (3.5), we get:

$$\begin{aligned} &p\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{a}'_i, \mathbf{a}'', \mathbf{r}_i\right) = \\ &\quad \Pr\left(\mathcal{L}_i^{\omega_i}\right) p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\omega_i}\right) \prod_{\forall j \neq i} p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\omega_i}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}''_j \mid \mathcal{L}_j^{\omega_j}\right) \Pr\left(\mathcal{L}_j^{\omega_j}\right) \end{aligned}$$

Using this in (3.4) we obtain the following expression for the conditional labelling probability:

$$\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}'_i, \mathbf{a}'', \mathbf{r}_i\right) = \frac{\Pr\left(\mathcal{L}_i^{\alpha}\right) Q\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr\left(\mathcal{L}_i^{\omega_i}\right) Q\left(\mathcal{L}_i^{\omega_i}\right)}$$

where

$$\begin{aligned} Q(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) \times \\ \sum_{\omega_1 \in \Omega_1} \cdots \sum_{\omega_{i-1} \in \Omega_{i-1}} \sum_{\omega_{i+1} \in \Omega_{i+1}} \cdots \sum_{\omega_N \in \Omega_N} \left\{ \prod_{\forall j \neq i} p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}''_j \mid \mathcal{L}_j^{\omega_j}\right) \Pr\left(\mathcal{L}_j^{\omega_j}\right) \right\} \end{aligned}$$

Each factor in the product in the above expression depends on the label of only one other object apart from the object O_i under consideration. We can therefore rearrange and simplify it, and express the conditional labelling probability formulae as:

$$Q(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j'' \mid \mathcal{L}_j^{\omega_j}\right) \Pr\left(\mathcal{L}_j^{\omega_j}\right) \tag{3.10}$$

$$\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i\right) = \frac{\Pr\left(\mathcal{L}_i^{\alpha}\right) Q\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr\left(\mathcal{L}_i^{\omega_i}\right) Q\left(\mathcal{L}_i^{\omega_i}\right)} \tag{3.11}$$

The quantity $Q(\mathcal{L}_i^{\alpha})$ is known as the *support function* for the match $\mathcal{L}_i^{\alpha}$; it encapsulates the information available that provides support for updating the prior probability $Pr(\mathcal{L}_i^{\alpha})$.

Thus (3.10) and (3.11) tell us how to express the match probabilities conditional on attributes and relations as a function of the three quantities referred to earlier:

- information about the attributes,
- information about the relations and
- the prior match probabilities

The label that gives the largest updated (or a posteriori) probability is then chosen as the correct label.

We can view (3.10) and (3.11) as an update rule, that updates the prior match probabilities by combining them with measurement information in the form of attributes and relations:

$$\Pr\left(\mathcal{L}_i^{\alpha}\right) \xrightarrow{\mathbf{a},\mathbf{r}} \Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}, \mathbf{r}\right)$$

We note however that the measurement information may have been considerably diluted by the various independence assumptions.

This completes the derivation of the object-centred MAP update rule. In the next chapter we discuss how iterating the rule might ensure a consistent labelling. In the following two chapters (Chapters 5 and 6), we discuss how the terms in the labelling rule should be evaluated.

Chapter 4

Iterating the update rule

In this chapter we explain why the object-centred approach to object labelling described in the previous chapter can generate solutions that are not globally consistent. We describe a heuristic method that can rectify this problem: this method uses the update rule of the previous chapter in a relaxation (or iterative) scheme to encourage convergence to a more consistent global labelling.

4.1 A drawback of the object-centred approach

In setting out the MAP rule in the form of (3.1) in the previous section, we were assuming that the labelling of each object was independent of the labelling of any of the other objects (the object-centred approach); that is, we attempt to maximise the posterior probability $\Pr(\mathcal{L}_i^{\alpha}|\mathbf{a},\mathbf{r})$ of the labelling of each object O_i individually. This is in contrast to the message-centred approach, in which one would attempt to maximise the joint posterior probability $\Pr(\mathcal{L}_1^{\alpha_1},\ldots,\mathcal{L}_N^{\alpha_N}|\mathbf{a},\mathbf{r})$. The effect of this approach was evident in the final formula (3.10, 3.11), in which only the prior probabilities, which in practice generally contain little information, were combined with the attributes and relations to build up evidence for a particular labelling. Thus although the evidence for some particular labelling might be strong based on the *relations* with some particular labellings of the remaining objects, the consequence of this object-centred approach is that no account was taken of whether these other labellings (that were used to provide this evidence) were in themselves good labellings. In other words, if the relations $\mathbf{r}_{ij}$ between objects O_i and O_j match the relations $\mathbf{R}_{\alpha\beta}$ between labels α_i and β_j, this will provide good support (according to (3.10)) for the labelling $\mathcal{L}_i^{\alpha}$, irrespective of the plausibility of the labelling $\mathcal{L}_j^{\beta}$ of O_j with β_j.

We present a simple illustration of this effect, in which the features are line segments:

(a) scene (b) model

The scene has been subjected to some unknown translation and rotation with respect to the model, and noise has been added (in the form of some small arbitrary rotations of the line segments). When the MAP updating rule of (3.10) and (3.11) was used, object O_1 was labelled with label α, whereas object O_2 was labelled with label γ. These two labellings are clearly inconsistent with each other. On the other hand, the labelling for each object *on its own* was completely reasonable. We can see that this conflict arises because the objects were labelled independently from each other — we have not made use of the fact that the labelling $\mathcal{L}_1^{\alpha}$ was the MAP labelling when deciding on the label for O_2. Incidentally, we should note that, although for example O_1 was labelled with label α, it should be fairly obvious in this example that the support for this labelling can not have been all that much greater than that for label β.

4.2 Why iterating the update rule might help

We might summarise the update rule of (3.10, 3.11) as follows: we have some set of prior labelling probabilities, whose values have been determined by some means unrelated to the measurements we are considering here. The rule then expresses how to "improve" these probabilities by incorporating new information in the form of attributes and relations. What we would ideally like to do, in order to generate a consistent solution, is somehow to "improve" the prior probabilities before we combine them with the attributes and relations.

An obvious heuristic step is therefore to apply the update rule a second time, this time using the results of the first iteration as the prior probabilities — after all, we have not indicated as yet how the prior probabilities should be obtained, and naturally we want to use the best ones available to us. It seems reasonable that "prior" probabilities obtained in this way would be more useful to us than those we used for the first iteration. We can then repeat the process as many times as is needed to meet some convergence or consistency criterion, progressively propagating information from node to node across the graph in the manner of a relaxation algorithm, so that the final best labelling will be selected by combining strong relations with plausible neighbouring labellings. We take on trust here that the iterative procedure *does* converge to a stable, consistent and unambiguous solution. However, extensive testing of the algorithm on a wide range of applications, together with work by other authors [55, 56] encourages us to believe that the asymptotic solutions are indeed consistent and unambiguous in the sense defined by Hummel and Zucker in [33].

The objection to this scheme might be expressed as follows: since our formula generates the value $\Pr(\mathcal{L}_i^{\alpha}|\mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i)$, the label probability *given the measurement information*, the implication is that the term $\Pr(\mathcal{L}_i^{\alpha})$ denotes some probability that does *not* already include this information. Putting it another way: if we are calculating probabilities dependent on a set of measurements by combining the measurements with a set of probabilities that already incorporates this information, the updated probabilities should be no different from the previous set. We reject this view, for two reasons:

- Our view is that the update rule indicates how to update a set of label probabilities; it does not give any indication of how the prior probabilities are to be obtained, implying that the best possible information should be used for this at all times.

- From Assumption 3.1. in the previous chapter, the update rule uses restricted sets of attributes and relations, so that on the first iteration the posterior probability being calculated is $\Pr(\mathcal{L}_i^{\alpha}|\mathbf{a}_i', \mathbf{a}'', \mathbf{r}_i)$. On each subsequent iteration, more of the remaining information is incorporated, so that on convergence, the posterior probability being calculated is $\Pr(\mathcal{L}_i^{\alpha}|\mathbf{a}, \mathbf{r})$. In other words, we would maintain that a *different* posterior probability is being calculated each time.

4.3 The relaxation rule

The above discussion suggests that it might be possible to improve the solution to the problem of labelling, as defined by (3.1), by combining (3.10) and (3.11) in a relaxation scheme. To implement such a scheme, in the (n+1)th iteration we use for the prior probabilities $\Pr(\mathcal{L}_i^{\alpha})$ the posterior probabilities that were calculated by the nth iteration (*c.f.* [33, 40]):

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i|\mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j''|\mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \quad (4.1)$$

$$\Pr^{(n+1)}\left(\mathcal{L}_i^{\alpha}\right) = \frac{\Pr^{(n)}\left(\mathcal{L}_i^{\alpha}\right) Q^{(n)}\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr^{(n)}\left(\mathcal{L}_i^{\omega_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\omega_i}\right)} \quad (4.2)$$

The relaxation scheme is initialised using the original update rule (3.10) and (3.11), *i.e.* by setting $\Pr^{(1)}(\mathcal{L}_i^{\alpha})$ to be the prior probability $\Pr(\mathcal{L}_i^{\alpha})$. When the algorithm terminates, we use equation 3.2 to determine the MAP labelling. The remaining issue is therefore to decide how many iterations are needed. A simple heuristic would be to terminate when a certain labelling is reached, that is when each object is assigned one label only with probability 1, the probabilities for all other labels for that particular object being zero. Since the form of updating rule that we have derived can only approach the state of unambiguous labelling asymptotically, in practice we must adopt some heuristic rule. Some possibilities are listed here:

- Since the algorithm in its iterated form is a relaxation algorithm, and noting the discussion in Section 4.2:

  (i) iterate enough times to ensure that all of the measurement information has been propagated to all of the nodes, then

  (ii) iterate once more to ensure that the relational information is combined with plausible labellings.

  This rule would suggest that, for applications in which the scene and model graphs are fully-connected (all possible relations are included), two iterations should be enough. As our results indicate later on (Chapter 9), we find that very little improvement was gained by iterating more than twice in most cases.

- Terminate if, for each scene node, one of the match probabilities exceeds $1 - \varepsilon_1$, where $\varepsilon_1 \ll 1$.

- Terminate if, during the current iteration, none of the probabilities changed by more than ε_2, where $\varepsilon_2 \ll 1$.
- Terminate if, during the current iteration, both of the following conditions are met:

  (i) the probability of the labelling selected by the MAP rule increased, and

  (ii) the probability of all of the other labellings decreased.
- For real-time applications, the algorithm should terminate after some fixed number of iterations.
- Terminate if none of the labellings are changed by the current iteration.

When the iterative rule was applied to the example in Section 4.1, with the termination criterion that each MAP labelling must have a probability of at least 0.9999, the algorithm terminated after 3 iterations. Of the two possible consistent labellings, the final labelling $\{\mathcal{L}_1^{\alpha}, \mathcal{L}_2^{\beta}\}$ was reached after the second iteration.

It is worth noting that the relation p.d.f. term $p\left(\mathbf{r}_{ij} | \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right)$ in the expression for the support function (4.1) plays a similar role to the compatibility coefficients of other relaxation methods (*e.g.* [33, 50]); that is, it quantifies the compatibility between the matches $\mathcal{L}_j^{\beta}$ and $\mathcal{L}_i^{\alpha}$. This relationship is explored further in Chapter 11.

4.4 Drawbacks of iteration

There are three important practical implications that result from the inclusion of iteration in the algorithm:

- Since the relation p.d.f.s are used in each iteration, they have either to be stored or to be recalculated on each iteration. If they are stored, this creates by far the largest demand for data storage in the implementation of the algorithm ($N^2(M+1)^2$ coefficients in the worst case), which placed a severe restriction on the size of matching problem that can be handled. If they are recalculated each time, the extra computation considerably increases the total computational load. On the other hand, for large problems, either the number of relations used or the number of labels that have to be calculated can usually be substantially reduced (see Chapter 7).
- More computation is required, since several iterations are needed. However if only a few iterations are needed, and the relation p.d.f.s are stored for the whole computation, the extra computation required for the iterations is a small fraction of the total.
- The amount of computation required is not known in advance, which might be a problem for a real-time implementation. In practice it is usually possible to put a (modest) upper limit on the number of iterations needed.

Chapter 5

Evaluation of the terms in the labelling formulae

The relaxation process of equations 4.1 and 4.2 requires the evaluation of three types of terms: the prior probabilities, which we describe first, and the p.d.f.s for the attributes and relations. Much of the detail of the p.d.f. evaluation is specific to the particular type of feature; in this chapter we discuss only the general aspects, leaving the more specific discussion to Chapter 6.

5.1 Assignment of the prior match probabilities

The prior (*i.e.* unconditional) match probability $\Pr(\mathcal{L}_i^{\alpha})$ represents the probability of the labelling $\mathcal{L}_i^{\alpha}$ in the absence of any measurement information. The estimation of these probabilities is usually a rather rough-and-ready affair, inevitably so since prior probabilities are by their nature quantities lacking in information. On the other hand, the method is not unduly sensitive to the values chosen.

We sometimes have some knowledge that an unconditional match probability involving the null node may be different from the others (typically it is larger); we denote this value by the constant ζ. Then, unless any evidence is available to the contrary, from the Principle of Indifference the prior probabilities of the remaining labellings for a given object are assumed to be equal to each other:

$$\Pr\left(\mathcal{L}_i^{\omega_i}\right) = \begin{cases} \zeta & \text{if } \omega_i = \emptyset \\ \frac{1-\zeta}{M_i} & \forall \omega_i \neq \emptyset \end{cases} \tag{5.1}$$

For example, if the number N of scene nodes is larger than the total number M of model nodes, it might be reasonable to suppose that around $N - M_i$ of the scene nodes will be labelled with the null model node. In this case we would put $\zeta = (N-M)/N$, and adjust the other prior probabilities according to (5.1). Alternatively we may know that, due say to poor image quality, we can estimate that some proportion of the scene features is likely to be spurious. We would therefore set ζ to this proportion.

If we have no information concerning the likely incidence of null labellings, we extend the Principle of Indifference and set ζ equal to the other prior match probabilities, *i.e.*

$$\forall \omega_i \ \Pr\left(\mathcal{L}_i^{\omega_i}\right) = \frac{1}{M_i + 1}$$

We illustrate the above argument in Fig. 5.1, in which the scene is taken from a greetings card and the model is the outline of a bird. The features in this example are line segments

(a) scene

(b) model

Figure 5.1: Greetings card scene and corresponding bird model with extracted line segment features

corresponding to edges. In this case, the scene is much larger than the model, and the task is to locate an instance of the model within the scene. The scene generated 88 features, and the model 6 features. Thus the null match prior probability was assigned a value of (88-6)/88.

5.2 Evaluation of the attribute p.d.f.

To evaluate the attribute p.d.f. $p(\mathbf{a}_i|\mathcal{L}_i^{\alpha})$, we use the attribute values as a set of arguments for the p.d.f. The problem that remains is therefore to determine the form of the p.d.f. itself. The p.d.f. can have one of two forms: the *null p.d.f.* $p(\mathbf{a}_i|\mathcal{L}_i^{\emptyset})$, when the label in question is the null label, and the *non-null p.d.f.* $p(\mathbf{a}_i|\mathcal{L}_i^{\omega_i\neq\emptyset})$, when the label in question corresponds to a physical model feature. We discuss each case in turn.

5.2.1 The attribute p.d.f. for non-null labellings

As was discussed in Section 2.4, the scene attributes are derived from the mapping of some of the scene measurements into the model measurement space. We assume that in practice the scene measurements will already have been transformed in such a way that the true, *i.e.* noiseless (but unknown) attribute values will be close to the corresponding attributes of the correct label attribute values. For example, if the original image and model were of different (but known) scales, we assume that the image (or model) is rescaled appropriately. We can then assume that the object attributes and those of the *correct* label differ only by the two types of error introduced respectively by the noise in the scene and the scene-to-model transformation uncertainties. In order to be able to evaluate the expression $p(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha})$, where α is some non-null label, we need to express these two types of error explicitly, using the total probability formula to incorporate the uncertain element of the transformation, $\Phi_{\mathbf{a}}$:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) = \int p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right) p\left(\Phi_{\mathbf{a}} \mid \mathcal{L}_i^{\alpha}\right)\, d\Phi_{\mathbf{a}} \tag{5.2}$$

In the absence of any measurement information, $\Phi_{\mathbf{a}}$ is independent of the match $\mathcal{L}_i^{\alpha}$; that is,

$$p\left(\Phi_{\mathbf{a}} \mid \mathcal{L}_i^{\alpha}\right) = p\left(\Phi_{\mathbf{a}}\right)$$

Hence (5.2) simplifies to:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) = \int p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right) p\left(\Phi_{\mathbf{a}}\right)\, d\Phi_{\mathbf{a}} \tag{5.3}$$

The first term, $p(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}})$, is a p.d.f. that describes the effect of the attributes given the transformation $\Phi_{\mathbf{a}}$; that is, it models the uncertainty in the scene attributes due to the measurement errors that result from the feature extraction process. The second term, $p(\Phi_{\mathbf{a}})$, models the uncertainty in the transformation from the scene to the model domains. We consider each of these terms in turn.

The influence of the attributes given the relationship between the frames of reference

We have a set of attributes $\mathbf{a}_i$ of object O_i. If label α is the correct label for O_i, and we are able to precisely map the scene onto the model using the transformation operator $\Phi_{\mathbf{a}}$, then the label attributes $\mathbf{A}_{\alpha}$ are the correct values for the object attributes $\mathbf{a}_i$, and will therefore be

the mean for their distribution. Thus because α is the *correct* label, the attribute p.d.f. will be some function of the form:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right) = \mathcal{F}\left[\Phi_{\mathbf{a}}(\mathbf{a}_i) - \mathbf{A}_{\alpha}\right]$$

where $\mathcal{F}(\mathbf{x})$ is typically some unimodal function with a maximum at $\mathbf{x} = 0$. For example, if the measurement error distribution is Gaussian, and the attributes are some linear combination of the measurements, then

$$\begin{aligned} p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right) &= \frac{1}{(2\pi)^{\frac{K_{\mathbf{a}}}{2}} |\Sigma|} \exp\left\{\frac{1}{2}\left[\Phi_{\mathbf{a}}(\mathbf{a}_i) - \mathbf{A}_{\alpha}\right]^T \Sigma^{-1}\left[\Phi_{\mathbf{a}}(\mathbf{a}_i) - \mathbf{A}_{\alpha}\right]\right\} \\ &= \mathcal{N}_{\Phi_{\mathbf{a}}(\mathbf{a}_i)}\left(\mathbf{A}_{\alpha},\, \Sigma\right) \end{aligned} \tag{5.4}$$

where $\mathcal{N}_{\mathbf{x}}(\bar{\mathbf{x}}, \Sigma)$ denotes a (multivariate) Gaussian distribution for the random variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariance matrix Σ.

If we consider now another label, β say, that is not the correct label, $\mathbf{A}_{\beta}$ will not be the correct value for the mean of the distribution. Because the p.d.f. is unimodal, with a maximum at the mean value, evaluating the p.d.f. with the correct label is highly likely to give a larger result than with an incorrect one. (A similar argument applies to the evaluation of the relation p.d.f.s.) This observation gives us confidence that the update rule of (3.10) and (3.11) in combination with the decision to use the MAP labelling decision rule of (3.1) is likely to give the required labelling.

The relationship between the scene and model frames of reference

Because $\Phi_{\mathbf{a}}$ is a constant, ideally it should be regarded as a parameter whose value should be estimated as part of the labelling process. However, for this labelling method, the preceding arguments indicate that we must regard it as another random variable, and we must therefore decide on its likely distribution in order to perform the integration in (5.3). Without knowledge of the application, we can only make a few general comments here.

In some applications we will know enough about the relationship between the coordinate systems in order to map the attributes precisely. For example, we might wish to match line segments in an application for which the relative orientation of scene and model is known. In this case there might be just one attribute type: the segment orientation. In such a case, the p.d.f. for the attribute transformation, $p(\Phi_{\mathbf{a}})$, collapses to a delta function, and (5.3) becomes (for non-null matches):

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right)$$

Alternatively we may only be able to estimate the mean and variance of $\Phi_{\mathbf{a}}$. In this case, from maximum entropy considerations, $p(\Phi_{\mathbf{a}})$ should take the form of a Gaussian distribution. If all that we know is that $\Phi_{\mathbf{a}}$ lies within some region $\mathcal{D}$ of volume D, then maximum entropy considerations indicate that a uniform distribution over that region is appropriate:

$$p(\Phi_{\mathbf{a}}) = \begin{cases} \frac{1}{D} & \text{if } \Phi_{\mathbf{a}} \in \mathcal{D} \\ 0 & \text{otherwise} \end{cases}$$

5.2.2 The attribute p.d.f. for null labellings

In previous methods that used null nodes [10], the attributes were simply discarded when the null node was used as the label. Our method requires that measurements are used for all labellings, so a distribution must be estimated for attributes that are labelled with the null label. There are two reasons why a null labelling might be generated. The model may be incomplete, in which case some model nodes will be missing and the null node provides an alternative label. Alternatively, if the image is noisy, we may find that the feature extraction process generates spurious nodes, which should therefore be labelled with the null model node. We consider these two situations in turn.

Incomplete model

In some applications, because the model is imperfect it may have some missing nodes. For example, in the stereo matching application described in Chapter 9 (where we designate one image to be the "scene" and the other the "model"), there may be an edge in the scene that is occluded in the model, or it may lie just outside the border of the model. In such a case, we take the view that the missing node does exist, but its attributes and relations with other nodes are unknown; this node therefore has the character of a "wild card", to which we can match all nodes in the scene whose genuine match is missing. That there is only one null model node presents no problem, because our theory explicitly permits one model node to label many object nodes. If we wish to explicitly reflect the existence of several null nodes, the prior probability of the null match should be adjusted accordingly (Section 5.1).

From this viewpoint, we consider that the measurements $\mathbf{A}_\emptyset$ for the null node have unknown values, so we can interpret them as a set of random variables. Therefore in order to determine the appropriate form of the p.d.f., we use the theorem of total probability to expand the p.d.f., making the measurements $\mathbf{A}_\emptyset$ explicit:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^\emptyset\right) = \int p\left(\mathbf{a}_i \mid \mathbf{A}_\emptyset, \mathcal{L}_i^\emptyset\right) p\left(\mathbf{A}_\emptyset \mid \mathcal{L}_i^\emptyset\right) d\mathbf{A}_\emptyset \tag{5.5}$$

This first term under the integral, $p(\mathbf{a}_i|\mathbf{A}_\emptyset, \mathcal{L}_i^\emptyset)$, is the conditional p.d.f. for $\mathbf{a}_i$ given the location of the missing null model node. It is therefore of the same form as the p.d.f. $p(\mathbf{a}_i|\mathcal{L}_i^\alpha)$ in the non-null case (in which, we may remember from Chapter 3, the model measurement $\mathbf{A}_{\omega_i}$ was included implicitly).

For the second term, $p(\mathbf{A}_\emptyset|\mathcal{L}_i^\emptyset)$, knowledge of the labelling provides no information without the object measurements; hence

$$p\left(\mathbf{A}_\emptyset \mid \mathcal{L}_i^\emptyset\right) = p\left(\mathbf{A}_\emptyset\right)$$

All we can say about the model measurements in general is that they must lie within the model domain. All we know about the set of measurements of the *correct* labels is that they must lie within the scene domain. For this term we again adopt the maximum entropy approach; thus $\mathbf{A}_\emptyset$ should be uniformly distributed over the smaller of the scene and model attribute domains, which we denote by the symbol $\mathcal{D}_\mathbf{a}{}'$, and which has a volume $D_\mathbf{a}{}'$. For

many applications, the extent of the model domain is not well defined, in which case the scene domain is used.

In practice, the second term $p(\mathbf{A}_\emptyset)$ will have a much larger variance than the first, $p(\mathbf{a}_i|\mathbf{A}_\emptyset, \mathcal{L}_i^\emptyset)$, and will therefore dominate the result; hence we put

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^\emptyset\right) = \frac{1}{D_\mathbf{a}{}'}, \ \forall \mathbf{a}_i \in \mathcal{D}_\mathbf{a}{}'$$

where $D_\mathbf{a}{}'$ is the size of $\mathcal{D}_\mathbf{a}$.

Spurious image nodes

Because the scene data is usually noisy, not only will this affect the measurements of genuine features extracted from the scene, but it also creates the possibility that spurious scene nodes may be generated that do not correspond to actual physical features at all. We cope with this situation by permitting such spurious nodes also to be labelled with the null model node. In this case, the null node has no physical existence, so it neither has attributes nor has relations with any other model node. Thus the conditional part of the expression for the attribute p.d.f. (*i.e.* the match $\mathcal{L}_i^\emptyset$ in the expression $p(\mathbf{a}_i|\mathcal{L}_i^\emptyset)$) tells us nothing about the attributes involving the spurious scene nodes, and hence provides no information towards evaluating the p.d.f. We therefore take the maximum entropy view again, and assume that in the absence of any other information the p.d.f. is uniformly distributed within the scene attribute domain $\mathcal{D}_\mathbf{a}{}''$. Hence

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^\emptyset\right) = \frac{1}{D_\mathbf{a}{}''}, \ \forall \mathbf{a}_i \in \mathcal{D}_\mathbf{a}{}''$$

where $D_\mathbf{a}{}''$ is the size of $\mathcal{D}_\mathbf{a}{}''$. This result is not necessarily the same as for the previous case of missing model nodes, depending on the relative extents of the scene and model domains. In a particular application, a view must be taken as to which is the more likely cause of null labellings. Fortunately the algorithm proves to be fairly insensitive to the value of the null match p.d.f. (see Section 9.2). The size of the actual domain used, $\mathcal{D}_\mathbf{a}$, is denoted by $D_\mathbf{a}$, so the attribute p.d.f. for a null match is given by:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^\emptyset\right) = \frac{1}{D_\mathbf{a}}, \ \forall \mathbf{a}_i \in \mathcal{D}_\mathbf{a} \tag{5.6}$$

5.2.3 Evaluation of the attribute domain size

In order to evaluate the null p.d.f., we have to estimate the domain size $D_\mathbf{a}$ of the attribute, *i.e.* the n-volume occupied by all the possible values that the multivariate attribute random variable can take. The method we employ here to find the n-volume is fairly crude: we take the product of the approximate ranges of the individual attribute types.

5.3 Evaluation of the relation p.d.f.

The reasoning behind the procedure for evaluating the relation p.d.f., $p(\mathbf{r}_{ij}|\mathcal{L}_i^\alpha, \mathcal{L}_j^\beta)$, is broadly the same as that for the attribute p.d.f. Again, the form of the p.d.f. depends on whether or not a null label is involved.

5.3.1 The relation p.d.f. when neither label is the null label

As before, the two types of error in the object measurements can be represented separately using the total probability formula:

$$p(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta}) = \int p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta},\Phi_{\mathbf{r}}\right) p\left(\Phi_{\mathbf{r}}\right)\, d\Phi_{\mathbf{r}} \tag{5.7}$$

When deciding which relation types are to be used, the aim is to choose a set that is invariant to the transformation $\Phi_{\mathbf{r}}$. Where this is the case, $p(\Phi_{\mathbf{r}})$ is a delta function and we can express the relation p.d.f. as:

$$p(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta}) = p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta},\Phi_{\mathbf{r}}\right)$$

By analogy with the attributes, the mean of the p.d.f. is given by the model relations. Thus if we were again to use the example where the relations are mutually independent and have a Gaussian distribution, the non-null p.d.f. would have the form:

$$p(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta}) = \mathcal{N}_{\mathbf{r}_{ij}}\left(\mathbf{R}_{\alpha\beta},\, \Sigma\right)$$

5.3.2 The relation p.d.f. when at least one label is null

In the case for which one label only is null, the remaining label on its own provides little useful information towards the form of the p.d.f. Thus this case is treated in the same way as that for which both labels are null. In both cases we refer to the p.d.f. as the null p.d.f. Using similar reasoning to that for the attributes, this p.d.f. is assumed to be uniform over the scene relation domain $\mathcal{D}_{\mathbf{r}}$, with value $1/D_{\mathbf{r}}$, *e.g.* :

$$p\left(\mathbf{r}_i \mid \mathcal{L}_i^{\alpha},\mathcal{L}_i^{\emptyset}\right) = \frac{1}{D_{\mathbf{r}}},\ \forall \mathbf{r}_i \in \mathcal{D}_{\mathbf{r}} \tag{5.8}$$

The relation domain size $D_{\mathbf{r}}$ is estimated in a similar fashion to that of the attributes. For example, consider an application in which points are to be matched, and there is an unknown translation. The relation set consists of an angle and a distance measurement, the angle has a range of 360°, and the distance measurement typically has a range R commensurate with the image size (the geometric mean of the width and height, say). Hence we estimate $D_{\mathbf{r}}$ to be $360R$.

5.3.3 Compatibility coefficients

We remarked earlier (Section 4.3) that the relation p.d.f. performs a role analogous to the compatibility coefficients of other relaxation methods. One of the differences is that the p.d.f. clearly is not dimensionless (its values have units that are the reciprocal of the random variable concerned), unlike the compatibility coefficients of Hummel and Zucker [33] for example. We also find it convenient to talk in terms of compatibility coefficients. We therefore normalise the p.d.f. values (without affecting the outcome of the update rule), by dividing by some neutral value. The null match density would appear to be an appropriate value for this: a null labelling $\mathcal{L}_j^{\beta}$ cannot express support either for or against a labelling

$\mathcal{L}_i^{\alpha}$ because there is no contextual information. Thus coefficient values greater than unity represent evidence that the matches are to some degree compatible, and values in the range $[0 \ldots 1)$ indicates some incompatibility. We note that taking the logarithm of these values would then generate coefficients of the form used by Hummel and Zucker.

5.4 A comment on p.d.f. evaluation

As was discussed in Chapter 3, each object is labelled independently of the others, and the assumption was also made that the attributes and relations are independent. This assumption has an unfortunate consequence when we come to evaluate the attribute and relation p.d.f.s using the formulae of (5.3) and (5.7). In these formulae, we remember that application of the total probability theorem uses the p.d.f. for the uncertain transformation, $p(\Phi)$. Now the transformation Φ applies to the whole scene; that is, there is in practice a strong correlation between the transformations applied to each individual object. Because of the independence assumptions, this fact is completely ignored, so that the transformation applied to each object is taken to be independent from all of the others, possibly entailing a serious loss of information.

To minimise the effect of this problem, we should therefore look for sets of attributes and relations that minimise dependence on Φ, so that the p.d.f. $p(\Phi)$ is as compact as possible. Indeed this is exactly the reason that we use relations as well as attributes: it is the ability to replace a set of attributes by some equivalent set of relations that gives us a chance of finding a superior distribution for Φ. Thus a set of relations (or attributes) that is invariant to a transformation is equivalent to a set of relations (attributes) for which the transformation p.d.f. $p(\Phi_{\mathbf{r}})$ (or $p(\Phi_{\mathbf{a}})$) is compact.

Chapter 6

Points and line segments as features

In this chapter we continue the discussion of the previous chapter on the evaluation of the attribute and relation p.d.f.s, in particular of those that do not depend on a null labelling. To do this, we must first decide on the types of attributes and relations to be used. This in turn will depend on the following:

- What actual geometric feature do the nodes on the graphs represent, and hence what measurements are associated with these features?
- What knowledge do we have of the relationship between scene and model spaces?

Having selected the sets of attributes and relations, we show how to use the results of Chapter 5 (equations (5.3) and (5.7) respectively) to evaluate the p.d.f.s. In the examples of this chapter, the uncertainty of the scene-to-model transformation and the measurement errors is assumed to be additive; thus the application of the uncertain operator Φ becomes the subtraction of the random variable Φ, and (5.3, 5.7) can be written as convolutions:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}, \Phi_{\mathbf{a}}\right) * p\left(\Phi_{\mathbf{a}}\right) \tag{6.1}$$

$$p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}\mathcal{L}_j^{\beta}\right) = p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}, \Phi_{\mathbf{r}}\right) * p\left(\Phi_{\mathbf{r}}\right) \tag{6.2}$$

We first discuss how the preceding theory may be applied to problems in which the features are points in 2-D space. We then extend the discussion to features that are straight line segments. In addition the segments may be undirected (*i.e.* they have a 180° ambiguity in their orientation) or directed (no such ambiguity).

6.1 Points as features

Points represent the simplest type of feature; such features may be generated for example by a corner- or junction-detection process. They have just two associated measurements, which identify their position in 2-D space. We assume that the errors from each feature are

independent of those of other features, in order to satisfy the independence assumptions in Chapter 3.

As was mentioned above (and was discussed in Section 2.5), the selection of attribute and relation types is determined by our knowledge of the relationship between scene and model spaces. We consider first an example in which there is an unknown translation between scene and model; that is, the relative position between scene and model planes is unknown but the relative scale and orientation are known. There are therefore no attributes, because we cannot relate the position of an isolated scene feature with that of an isolated model feature in any way, except to say that (presumably) the label for the scene feature lies *somewhere* within the model. Thus the second term under the integral in (6.1), $p(\Phi_{\mathbf{r}})$, is a uniform p.d.f. over the whole model attribute space, and we can say that:

$$p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) \approx \frac{1}{D_{\mathbf{a}}}$$

(Because it is constant, this term will cancel out in the update rule of (3.10)).

On the other hand there will be two relation types, specifying the position of one node relative to another. We could use for example a Cartesian representation, so that each object d_i is represented by the measurement pair $\mathbf{x}_i = (x_i, y_i)$. The relations would then be:

$$\begin{aligned} r_{ij}^{(1)} &= x_j - x_i \\ r_{ij}^{(2)} &= y_j - y_i \end{aligned}$$

The second term under the integral in (6.2), $p(\Phi_{\mathbf{r}})$, is a delta function, so that

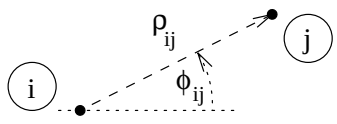
$$p\left(\mathbf{r}_i \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right) = p\left(\mathbf{r}_i \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}, \Phi_{\mathbf{r}}\right)$$

Hence to evaluate the relation p.d.f. we need to know the distribution of the image noise, which must be estimated from knowledge of the feature detection process. If for example we know that the measurement errors for a feature are uncorrelated (which is normally the case for point features), and have a Gaussian distribution with variances σ, it follows that:

$$p\left(\mathbf{r}_i \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right) = \mathcal{N}_{\mathbf{x}_j - \mathbf{x}_i}\left\{\mathbf{X}_{\beta} - \mathbf{X}_{\alpha}, \begin{pmatrix} 2\sigma & 0 \\ 0 & 2\sigma \end{pmatrix}\right\}$$

Next we consider some variations on the above example.

Scene orientation unknown: If the scene orientation is unknown, the transformation p.d.f. $p(\Phi_{\mathbf{r}})$ is no longer a delta function. If the Cartesian representation of the relations were to be retained, $p(\Phi_{\mathbf{r}})$ would be difficult to determine. Instead we use a polar representation centred on O_i; the relations $\mathbf{r}_{ij}$ are then represented by the distance ρ and relative orientation θ of O_j from O_i (and corresponding relations P, Θ for the model):



If the measurement noise variance is small, the p.d.f. has two independent components: one in the radial direction, which remains a delta function, and one in the angular direction, which is uniform over the whole angular range. Therefore the angular measurements no longer convey any useful information, and we retain only the distance between the nodes as a relation:

$$r_{ij}^{(1)} = \rho_{ij}$$

The result for the Gaussian example would then be:

$$p\left(\mathbf{r}_i \,|\, \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right) = \mathcal{N}_{\rho_{ij}}\left(P_{\alpha\beta},\, 2\sigma\right)$$

Small unknown error in scale: In this case, using a polar representation again, the angular component of $p(\Phi)$ is a delta function. If the scaling error is expressed as an unknown scale factor $s = 1+\delta$, where δ is uniformly distributed in the range $|\delta| < \varepsilon$, the relations are:

$$\begin{aligned} r_{ij}^{(1)} &= \rho_{ij} + \delta\rho_{ij} \\ r_{ij}^{(2)} &= \theta_{ij} \end{aligned}$$

Thus the distance component ϕ_d of $\Phi_{\mathbf{r}}$ is here the quantity $\delta\rho$, which (ignoring the errors in ρ) will have a uniform distribution with range proportional to the distance in question, *i.e.*

$$p(\phi_d) = \frac{1}{2\varepsilon\rho}, \quad -\varepsilon\rho < \phi_d < \varepsilon\rho$$

Some observations of a practical note are in order. Firstly the two distributions, $p(\mathbf{r}_i|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}, \Phi)$ and $p(\phi_d)$, have to be convolved in order to evaluate the relations. If they were for example Gaussian and uniform distributions respectively, the convolution integral would be tedious to evaluate. It is usually the case that the form of the former distribution can be calculated, whereas that of the latter is often estimated heuristically; it may therefore be convenient to use a distribution for $p(\phi_d)$ that makes the integral easier to do. For example, if $p(\mathbf{r}_i|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}, \Phi)$ has a Gaussian distribution, it would simplify matters to make the distribution for $p(\phi_d)$ Gaussian also, with standard deviation $\varepsilon\rho$.

Secondly, because the width of the distribution varies with the distance between the features, for distant pairs of features the distribution becomes relatively broad, so that the information provided by the relation will become correspondingly less useful. Since the occurrence of pairs with a given separation increases with the separation, useful computational savings may sometimes be made by omitting the calculations involving distant feature pairs. This is discussed in more detail in Section 7.2.2.

Positional information as attributes: In some applications, *e.g.* stereo matching, it is often the case that some positional information relating the image and model is available. For example, a feature at a given height in one image is likely to be at a similar height in the other; also there will be some horizontal correlation. This information can be included as pairs of positional attributes. The distribution used for these attributes is likely to be broader than those of the relations, particularly in the horizontal direction. The relations are therefore still included, even though it might seem that the same information is included twice.

The measurement information is available in a stronger form as relations; on the other hand, as attributes the information can be used to prune out improbable labellings from the calculation at the outset (see Section 7.3).

In such an application, the distributions for Φ will have to be estimated from knowledge of the camera separation and the likely distance of the features from the cameras.

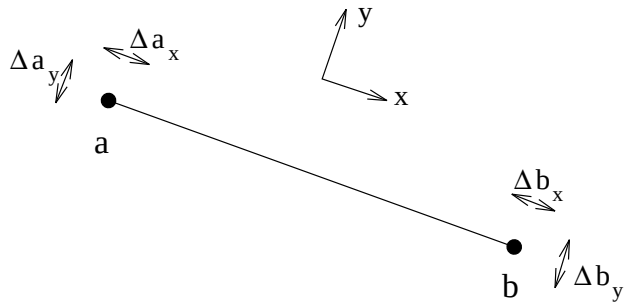
6.2 Line segments as features

The other feature type we consider in this chapter is the line segment. Line segments are typically created from an edge detection process. Edge detectors usually consist of several stages: an image is filtered to enhance the edges; the enhanced image is differentiated in two orthogonal directions, and the maxima of the resulting images are detected. The pixels corresponding to the maxima are linked into strings, and polygons are then fitted to the strings. The edges of the polygons are the straight line segments that are used as features for the matching process. In other applications (*e.g.* road matching), intensity ridges may be detected, and approximated as a set of line segments in the same way.

A line segment is a more complicated type of feature than a point, and the derivation of the relations and their distributions is correspondingly less straightforward. We begin by defining which measurements are used (Section 6.2.1). Then, using the case of an unknown Euclidean transformation, we choose an appropriate set of attributes and relations (Section 6.2.2), with their corresponding variances and covariances. As was done for the point features, we discuss what attributes and relations we need for other types of unknown transformation (Section 6.2.3). Finally (Section 6.2.5) we distinguish between directed and undirected line segments.

6.2.1 Measurements for line segments

A line segment is defined by four measurements: these are usually the positions in 2-D space of its two endpoints *a* and *b*. Using a Cartesian representation local to the segment and aligned with it, we define errors for each measurement:



In what follows, we make the following simplifying assumptions:

1. The error in the position of a segment endpoint can be characterised by two independent distributions, one collinear and one perpendicular to the segment; the respective variances are denoted as σ_{xx} and σ_{yy}.

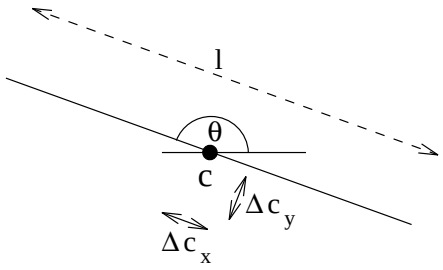
2. The errors of the two endpoints of a segment are independent.

3. The errors of a segment are independent of those of other segments (*c.f.* point features).

The last assumption is not always valid; for example two neighbouring segments could have a common endpoint. However such cases are the the exception rather than the rule.

To simplify the notation, we also assume in this section that the distributions of the endpoint errors are constant throughout the problem. It is a minor additional complication to deal with cases in which this is not true.

When deriving the relations, it is more convenient to represent the measurements as the position of the segment centre point $\mathbf{c}$ and its length l and global orientation θ. The errors $(\Delta c_x, \Delta c_y)$ are aligned with the segment as before:



Thus, assuming that the measurement errors are small compared to the segment length:

$$\begin{aligned} \Delta c_x &= \frac{\Delta b_x + \Delta a_x}{2} \\ \Delta c_y &= \frac{\Delta b_y + \Delta a_y}{2} \\ \Delta\theta &= \frac{\Delta b_y - \Delta a_y}{l} \\ \Delta l &= \Delta b_x - \Delta a_x \end{aligned}$$

The corresponding measurement error variances are then given by:

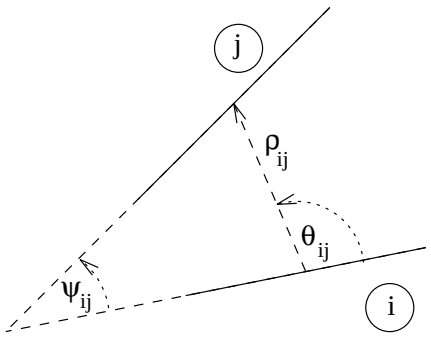
$$\begin{aligned} \sigma_{c_x c_x} &= \frac{\sigma_{xx}}{2} \\ \sigma_{c_y c_y} &= \frac{\sigma_{yy}}{2} \\ \sigma_{\theta\theta} &= \frac{2\sigma_{yy}}{l^2} \\ \sigma_{ll} &= 2\sigma_{xx} \\ \sigma_{c_x c_y} &= \sigma_{c_x\theta} = \sigma_{c_x l} = \sigma_{c_y\theta} = \sigma_{c_y l} = \sigma_{\theta l} = 0 \end{aligned}$$

Because of the way that the segments are generated, the segment length l turns out to be less useful than the other three measurements, mainly because in practice lines are often broken

in an arbitrary manner. As a result we tend to avoid using the segment length to generate attributes and relations, although it is still needed in estimating the relation variances. This is discussed further in Section 6.3.

6.2.2 Attributes and relation types for an unknown Euclidean transformation

Initially we consider an unknown Euclidean transformation between scene and model (unknown relative position and orientation but known relative scale). In this case none of the three (useful) measurement types can be used as attributes, but we can use them to define three relations that are invariant to the transformation. We chose three that were simple to express: the angle between the segments ψ, and the position of the centre of one segment with respect to that of the other expressed in a local polar representation ρ, θ:



Other choices are possible; for example a local Cartesian representation could have been used for the relative centre position in place of the polar one.

Because the relations are invariant to the transformation, Φ_r is a delta function, and hence

$$p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right) = p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}, \Phi_{\mathbf{r}}\right)$$

The errors in the relations can be expressed in terms of the measurement errors:

$$\begin{aligned}
\Delta\rho_{ij} &= -\cos\theta_{ij}\Delta c_{i_x} - \sin\theta_{ij}\Delta c_{i_y} - \cos\theta_{ji}\Delta c_{j_x} - \sin\theta_{ji}\Delta c_{j_y} \\
\Delta\theta_{ij} &= \frac{1}{\rho_{ij}}\left(\sin\theta_{ij}\Delta c_{i_x} - \cos\theta_{ij}\Delta c_{i_y} + \sin\theta_{ji}\Delta c_{j_x} - \cos\theta_{ji}\Delta c_{j_y}\right) - \Delta\theta_i \\
\Delta\psi_{ij} &= \Delta\theta_j - \Delta\theta_i
\end{aligned}$$

Although we have not specified the distributions of the measurement errors, we can exploit the independence assumptions in Section 6.2.1 to calculate the variances and covariances for the relation errors:

$$\begin{aligned}
\sigma_{\rho_{ij}\rho_{ij}} &= \left(\cos^2\theta_{ij} + \cos^2\theta_{ji}\right)\frac{\sigma_{xx}}{2} + \left(\sin^2\theta_{ij} + \sin^2\theta_{ji}\right)\frac{\sigma_{yy}}{2} \\
\sigma_{\theta_{ij}\theta_{ij}} &= \frac{1}{2{\rho_{ij}}^2}\left[\left(\sin^2\theta_{ij} + \sin^2\theta_{ji}\right)\sigma_{xx} + \left(\cos^2\theta_{ij} + \cos^2\theta_{ji}\right)\sigma_{yy}\right] + \frac{2\sigma_{yy}}{{l_i}^2} \\
\sigma_{\psi_{ij}\psi_{ij}} &= 2\left(\frac{1}{{l_i}^2} + \frac{1}{{l_j}^2}\right)\sigma_{yy}
\end{aligned}$$

$$\begin{aligned}
\sigma_{\rho_{ij}\theta_{ij}} &= \frac{1}{4\rho_{ij}}\left(\sin 2\theta_{ij} + \sin 2\theta_{ji}\right)\left(\sigma_{yy} - \sigma_{xx}\right) \\
\sigma_{\rho_{ij}\psi_{ij}} &= 0 \\
\sigma_{\theta_{ij}\psi_{ij}} &= \frac{2\sigma_{yy}}{{l_i}^2}
\end{aligned}$$

In earlier work (*e.g.* [13]), it was assumed that the errors in the relations had variances that were constant over the set of objects. It is apparent from the above expressions that, since several of the covariance matrix terms are functions of the relations themselves, this assumption was an oversimplification. It was also assumed that the covariance matrix was diagonal, which is clearly not the case for the particular set of relations we have chosen.

6.2.3 Other types of unknown transformation

There are many different types of unknown transformation that can be handled in a similar way to the Euclidean example above. We give two examples, one in which more information is available, and one in which there is less.

Unknown translation: In this case, because the relative orientation of scene and model is known, the segment orientation can be used as an attribute. The relation ψ (the angle between the segments) is then redundant, and is therefore no longer used.

Unknown Euclidean transformation with small scaling error: Although the unknown transformation is nominally Euclidean, in practice *small* scaling errors or other distortions are often encountered. In order to generate a set of relations that are invariant to scaling errors, we have two options:

(i) discard the distance relation ρ entirely, which entails a loss of a substantial proportion of the structural information, or

(ii) use ternary relations, an option that was rejected from the outset (p. 8).

In practice however, as in the case of point features, small amounts of scale distortion may be accommodated by adding an extra error term to the distance relations that is proportional to the distance itself. The only variance term that is affected is that for $\sigma_{\rho_{ij}\rho_{ij}}$, which (by analogy with that for the point features, p. 33) becomes:

$$\sigma_{\rho_{ij}\rho_{ij}} = \left(\cos^2\theta_{ij} + \cos^2\theta_{ji}\right)\frac{\sigma_{xx}}{2} + \left(\sin^2\theta_{ij} + \sin^2\theta_{ji}\right)\frac{\sigma_{yy}}{2} + (\varepsilon\rho_{ij})^2$$

where as before ε^2 is the variance of the scale error. The effect of this adjustment for the scaling error is to give emphasis to relations of nearby nodes in contrast to those of more distant nodes.

We remember however from the previous chapter (Section 5.4) that this solution suffers from the drawback that the scaling error is being modelled as a *separate* random variable for each

relation. This is clearly not true, and means that we are ignoring the fact that the scaling error is a single constant. On the other hand, *because* the scale factor error is not modelled as a single random variable, we can also use this method to model an arbitrary unknown deformation, provided that the amount of deformation increases with distance.

There are other systematic errors which we may also wish to model in a similar way. Consider for example an image of some structure for which we have a 3-D model: we wish to match corresponding features from the image and model. If the position of the camera is known approximately, the model can be projected into the image plane, with some error. Using knowledge of the projection process, extra terms can be added to the variances in a similar manner to that for the scaling error. For stereo matching, the epipolar constraint can similarly be used.

6.2.4 Unsystematic errors

In an application such as stereo image matching (described in Section 9.3), the unknown transformation between the images can be very roughly modelled as a translation. However this is a poor representation of the transformation, as the amount of translation for any feature pair will depend on the depth of the scene at that point. For this type of application it is not worth using the noise model for the relations derived in Section 6.2.2; instead we use a simpler model, for which the variances for the relation types are independent and constant throughout. The difficulty with this approach is that values for the variances must be estimated somehow.

Again, when finding the correspondence between an image and the projection of a 2-D model (Section 9.4), the assumption is made that the unknown transformation is Euclidean. A better uncertainty model could be found by assuming that the pose of the 3-D model is subject to errors, and explicitly propagating these errors through the projection process.

6.2.5 Directed and undirected line segments

A comment is in order concerning angle measurements in relation to line segments. A segment is defined by its pair of endpoints. In some applications the endpoints are ordered, *i.e.* we know which way round the segment is. In the case of segments derived from edge detection for example, the direction of the segments can be defined such that, if one looks along this direction, the intensity gradient of the image is positive towards (say) the right-hand side. These segments are *directed* segments; for them the segment orientation will be in the range $(-180^\circ \ldots +180^\circ]$, and arithmetic involving the orientation will be performed modulo 360°.

On the other hand, if we consider for example segments extracted by the detection of intensity ridges, or segments extracted from outline drawings, we do not know which way round the segment is, so the endpoints are not ordered, and the segment orientation will have an ambiguity of 180°. The segment orientation will therefore be in the range $(-90^\circ \ldots +90^\circ]$, and arithmetic on this measurement will be performed modulo 180°. Such a segment is an *undirected* segment. Less information is available as a result of the ambiguity, although some of it may be retrieved using a modification to the matching algorithm (Section 8.2).

There will also be cases for which both directed and undirected segments exist. For example, the scene may generate directed segments from an edge detection process. In such a case, the direction of the segment depends on the relative image intensity on either side of the edge. On the other hand, the model may well not have this information, and undirected segments will be generated. In such a case, the angle arithmetic will have to be performed modulo 180°.

6.3 Estimating the measurement errors in practice

In Sections 6.1 and 6.2, it was shown that the distributions of the attributes and relations could be expressed in terms of the measurement distributions, which in turn were determined by one or two variance terms, providing that the form of the measurement error distributions are known or can be estimated. In this section we discuss some possible origins of the measurement errors, which in turn gives an indication of how to assign values to the variances. In practice these variance values may only be approximate; however the method is not unduly sensitive to the actual values. We discuss the errors in terms of the variances σ_{xx} and σ_{yy} of the errors of a segment endpoint (defined in Section 6.2.1); for point features, the single error variance σ (Section 6.1) can be deduced by the same means as for σ_{yy}.

The matching algorithm has been evaluated on a number of different applications, for which the features were variously directed and undirected line segments and points. Several distinct sources of positional errors in the feature measurements were identified:

1. For all of the feature types, noise in the image creates errors, usually quite small.
2. For line segments, errors were caused by uncertainties in the collinear location of the segment end points, mainly due to line breakage. Line breakage occurs for different reasons in different applications:
   (a) Although the feature was genuinely a straight line, image noise and feature occlusion sometimes caused the feature detection process to break the segment into smaller ones. Occlusions are also caused by the image boundary.
   (b) The underlying feature was a curve that was approximated by a set of straight line segments in both scene and model: here the position of the segment endpoints along the curve is often arbitrary, and does not in general correspond between scene and model.
3. Because of the above causes of line breakage, the line features may sometimes be deliberately further broken into smaller pieces in order to give them a better-defined physical location.

The actual noise parameters for errors of type 1 should be found from an analysis of the processes that generate the segments from the original image; such an analysis can be found for example in [21], which provides a means for estimating σ_{xx} and σ_{yy} directly.

Errors of type 2 and 3 essentially affect only σ_{xx}. In practice the effect of line breakage is that the location of segment endpoints in a direction collinear with the segment is poorly defined,

resulting in a large value of σ_{xx} that swamps the effect of errors of type 1. If the errors are assumed to be uniformly distributed over a range of $\pm l/2$, σ_{xx} should be set to a value of $l^2/12$. Note that, if this approach is taken, the value of σ_{xx} is clearly different for different segments, especially for errors of type 2; the covariance expressions of Section 6.2.2 would therefore have to be modified accordingly.

For errors of type 2, since the correspondence of segment lengths between scene and model is poor (*i.e.* Δl is very large), the segment length provides little useful information. For errors of type 3, the segment lengths are fairly well-defined, but tend to be very similar; thus again they contain little information. Either way, because of these practical considerations we avoid using the segment length to generate attributes and relations. However, as we have seen, it is still needed in estimating the relation variances.

Chapter 7

Real-time issues

In many practical machine vision problems, the time taken for the algorithm to be computed is important. Storage requirements of the algorithm also have an impact on real-time performance, since in general larger data stores have slower access times. Sometimes there is a strict upper limit on the time available for computation — such applications are often described as "hard" real-time. We do not mind how long the program takes, provided this criterion is met. On the other hand sometimes there is no hard and fast limit, but there may be some rough time limit beyond which the performance would be regarded as unacceptable — a "soft" real-time application. Most software in the end has some requirement of this type, as there is usually a limit to the patience of even the most patient computer user. Since the basic iterative rule described in the previous chapters can be expensive in terms of both computation time and storage requirements, we need to examine the extent of these requirements, and then look at ways in which the computational load can be reduced in practice.

As discussed in Chapter 11, we use the term "compatibility coefficient" to denote a normalised relation p.d.f. Where the distinction is not important we use the terms interchangeably.

7.1 Identifying the problem

The iterated MAP update rule, derived in Chapters 3 and 4, is summarised overleaf:

iterate from $n = 1$ {

$\forall$ objects d_i, $i \in \{1 \ldots N\}$ $\forall$ labels $\omega_i \in \Omega_i$ {

$$Q^{(n)}(\mathcal{L}_i^{\omega_i}) = p\left(\mathbf{a}_i | \mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij} | \mathcal{L}_i^{\omega_i}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j'' | \mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \tag{7.1}$$

$$\Pr^{(n+1)}\left(\mathcal{L}_i^{\omega_i}\right) = \frac{\Pr^{(n)}\left(\mathcal{L}_i^{\omega_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\omega_i}\right)}{\sum_{\upsilon_i \in \Omega_i} \Pr^{(n)}\left(\mathcal{L}_i^{\upsilon_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\upsilon_i}\right)} \tag{7.2}$$

}

}

From this we can see that the most frequently-executed part of the algorithm is the multiply/accumulate operation that constitutes the inner-most loop of the calculation of the support functions. There are two parts of this computation that can dominate the processing time: the multiply/accumulate itself, and the calculation of the relation p.d.f.s. Thus both parts require processing power that is directly related to the number of compatibility coefficients. Since it takes significantly longer to compute a compatibility coefficient than to perform a multiply/accumulate, and since the coefficients are reused at each iteration of the relaxation process, it is highly desirable to store them (assuming that more than one iteration is required). This constitutes the main storage requirement for the algorithm; thus the storage requirement is also directly related to the number of compatibility coefficients.

We can see therefore that the key to reducing the computational requirements is to reduce the number of compatibility coefficients. As we have said, the coefficients are the same for each iteration, so there are four remaining loops, and the problem is one of fourth order. More precisely, the number $N_{\mathbf{r}}$ of non-trivial coefficients to be calculated is approximately the square of the total number of labellings, *i.e.*

$$N_{\mathbf{r}} \approx \left(\sum_{i=1}^{N} M_i\right)^2$$

where M_i is the number of possible non-null labels for object d_i. These coefficients can be considered as a four-dimensional array, which it is convenient to portray in the form of an array of sub-arrays (Fig. 7.1). There are N^2 sub-arrays, each of which corresponds to a different (ordered) pair of scene nodes, and each sub-array has M_i^2 elements. There are $\sum_i M_i$ rows and columns in the entire array. We can see that this ordering of the coefficients also lends itself to an efficient indexing arrangement for performing the multiply/accumulate operations in the expression for the support function (7.1).

We have identified three approaches by which we may be able to reduce the number of coefficient calculations. We can:

- identify relations containing little information that can be dealt with *en masse* (in effect, pruning the scene and model graphs),

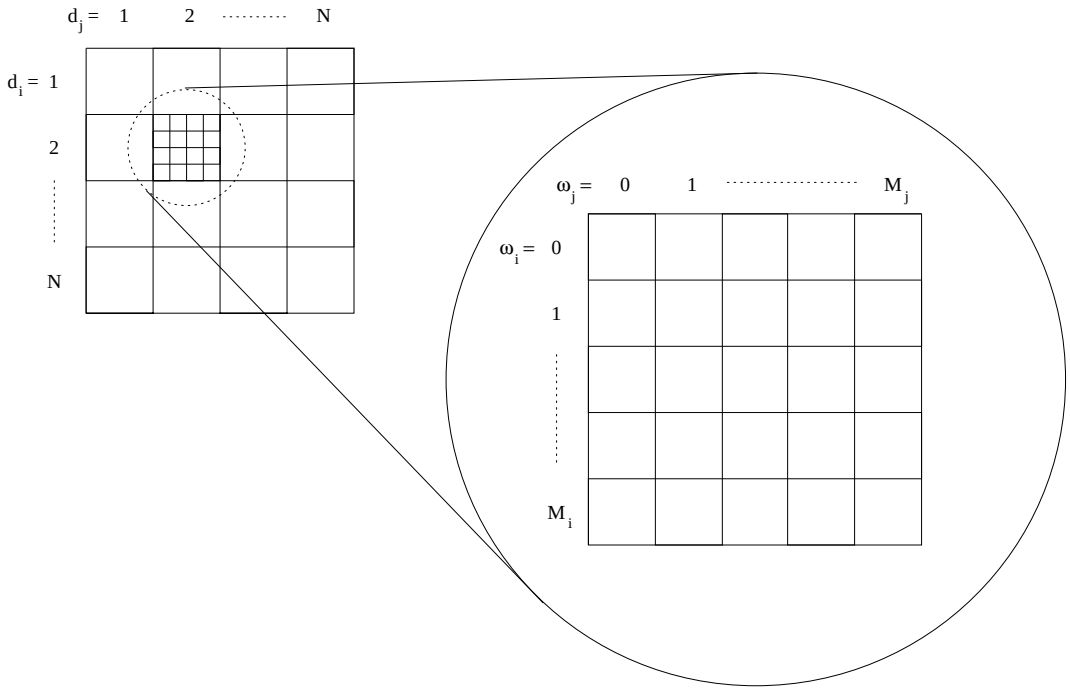


Figure 7.1: Arrangement of coefficients in a 2-D array

- reduce the total number of possible labellings, and
- reduce the number of features to be matched (initially, at least).

We consider each approach in turn. Before we do, though, one point should be stressed in relation to hard real-time applications. The basic algorithm described above has an amount of computation that can be precisely determined in advance if the sizes of the scene and model feature sets and the number of iterations are known. In practice, the number of iterations needed can be fixed in advance, so that the time needed to compute the matches can indeed be predetermined. This property is an essential requirement for a "hard" real-time application. Apart from the multiprocessing option, the methods described below all suffer from the same defect: the computation time can no longer be determined in advance, although the execution time of the unmodified algorithm still serves as an upper bound.

7.2 Identifying degenerate relations

7.2.1 Eliminating impossible pairs of nodes

Consider an application in which scene covers a much larger area than the model, for example when we are attempting to locate an object in a cluttered scene (*e.g.* the example of Section 5.1, discussed also in [15]). There will be many pairs of nodes in the scene for which there cannot possibly be a corresponding pair in the model; that is, at least one of the labels must be the null label. Each of these scene pairs corresponds to a complete sub-array in

Fig. 7.1. The values of the subarray terms are therefore all zero, with the exception of the right-hand column and the bottom row; the latter are the ones that involve the null label and therefore have the value of unity (*i.e.* the p.d.f. value is $1/D_{\mathbf{r}}$). The consequences of this are threefold:

- The compatibility coefficients do not have to be individually calculated.
- If storage for each individual sub-array is allocated separately, no storage need be allocated for these sub-arrays.
- Apart from the bottom row of the subarray, the inner, multiply/accumulate loop in the support function expression (7.1) consists of a single non-zero term, obviating the need to perform the multiply/accumulate for the whole row. For the bottom row, some computational savings are also possible.

We can adopt a similar strategy in the case of applications for which the model covers a much larger area than the scene, such as the example described in Chapter 1. However in this case it is the model which will contain pairs for nodes for which there are no possible matches in the scene. In this case, if we wish to economise on coefficient storage, it will be necessary to change round the storage strategy, so that each sub-array corresponds to a particular ordered pair of model nodes. Also the inner loop no longer processes consecutive zero-valued coefficients, so the computational savings are not so significant.

7.2.2 Restricting the range over which relations are included

Since we consider all possible pairs of nodes in the scene and model, it follows that, provided that a sufficiently rich set of relation types is used, there may be a large amount of redundancy in the set of compatibility coefficients. That is, if the relations between nodes O_i and O_j are sufficient to define node O_j completely with respect to O_i, and similarly for nodes O_i and O_k, then it follows that node O_k would be defined completely with respect to O_j. In practice our information is not perfect, and also we recognise that having the information implicitly is not the same as having it available explicitly. However we can still restrict the number of nodes that a given node has relations with, for example by only considering node pairs separated by less than some given distance.

The mechanics of this scheme operate in a similar way to those of the previous one (Section 7.2.1); the difference is that, for the node pairs that are excluded, the values of all of the corresponding compatibility coefficients are set to unity, indicating that we have no view as to whether the two possible matches concerned provide support for each other or not. We can therefore make the same economies in calculating and storing the coefficients as in Section 7.2.1; on the other hand, instead of omitting the corresponding multiply/accumulate operations, we have to replace them with accumulate operations.

This scheme is particularly appropriate for the situation in which we assume that the relative scale of scene and model is known, but which contains some small error. In this case, as was discussed in Chapter 6, the variance used for the distance relations will be small for node pairs that were close, but which would become increasingly large for node pairs that were

further apart. Once the node pairs are sufficiently far apart for the relation distribution to be dominated by the scaling error, the corresponding compatibility coefficients can be set to unity.

7.3 Pruning the label sets

When first analysing an application, usually the N label sets $\Omega_i, \forall i \in \{1 \ldots N\}$, are considered to contain the same M members; that is, each object can be labelled with any model label, and the total number of coefficients to be calculated is therefore $\approx (NM)^2$. However if some of the measurements can be used as attributes in the application, and the attribute p.d.f. is bounded, then it may be possible to eliminate some of the labellings from the outset. Thus for some labelling $\mathcal{L}_i^\alpha$, it may be the case that $p(\mathbf{a}_i|\mathcal{L}_i^\alpha) = 0$. This means that the labelling $\mathcal{L}_i^\alpha$ is impossible, so that we can remove the label α from the label set Ω_i from the outset. This corresponds to the deletion of a row and column from the array of Fig. 7.1.

For example we might know that the projection of the scene onto the model is Euclidean, and that the relative orientation of the scene is known to within a limit of $\pm\mu$. Also let us assume that the scene noise is such that the uncertainty due to noise of the orientation of any individual scene node is within say $\pm\nu$. Then the attribute p.d.f. lies entirely within the domain $[-\mu-\nu, \ldots, +\mu+\nu]$, and so all of the possible matches whose relative orientations differ by more than $|\mu+\nu|$ can be ignored.

7.4 An example

We tested the strategies of Sections 7.2 and 7.3 using the road-matching application described in the Introduction. In the example used here (Fig. 7.2) segments longer than 8 pixels were used: there were 13 segments in the scene, which was 60 pixels square, and the map, which was scaled to match the scene, contained 161 segments and covered an area of about 10 times that of the scene (it is about 190 pixels square). Matched segments are shown as black lines, and unmatched segments in the map are shown in grey. The matched segments were used to calculate the position and orientation of the image with respect to the map; this is represented in Fig. 7.2 by the correspondence of the cross and pointer symbols in the scene and map.

We ran the matching algorithm initially without any of the enhancement schemes; 4043676 (non-null) compatibility coefficients were computed, and the algorithm converged in one iteration,[1] taking 46s of c.p.u. time on a Sun Sparc10. Approximately 91% of the execution time was spent calculating the coefficients, and 5% performing the single relaxation step.

We next examine the effect on the algorithm of applying each of the three strategies described above. They were first tested separately, and then combined.

[1]The criterion for convergence was that one labelling for each node should have a probability of at least 0.9999.

(a) image (b) map

Figure 7.2: Matching of road segments

Eliminating impossible pairs of nodes

Since the model is larger than the scene, we will be searching here for pairs of model nodes which cannot simultaneously label any pair of scene nodes. The largest likely separation between pairs of model segments was estimated by finding the largest separation between pairs of scene segments, and adding the corresponding standard deviation (52 pixels in total). All possible pairs of model segments outside this range were then eliminated from the calculation as described above. In this case 869700 coefficients were computed. The algorithm still converged in one iteration, taking 14s to complete, and the same labelling was generated as before.

Restricting the range over which relations are included

For this strategy we specified the maximum distance between segments in the model over which relations would be considered. We set this distance to 30 pixels, with the result that 489996 coefficients were computed. Two iterations were required to reach convergence, and the algorithm took 10s to run. This time for three of the nodes the labelling was different; in each case there were two reasonable candidate labels, and the alternative was picked.

We can restrict the range of distances between segments in the scene; however when we tested this, four iterations were needed, with the result that the execution time was increased to 12s.

Pruning the label sets

In this case we assumed that the orientation of the map with respect to the scene is known to within $\pm 30^\circ$; this enabled us to restrict the set of possible labellings to those in which the segment orientations match to within this range. The number of potential matches was reduced from 2106 to 688; 419184 coefficients were generated, with an execution time of 5s.

Combining the schemes

When we combined the three schemes for reducing the computation, the number of coefficients computed was 43832 and the computation time 1.7s. The same labellings were generated as for the original example.

All of the above results are presented in Table 7.1 for ease of comparison.

scheme for reducing no. of coefficients	no. of coefficients	no. of iterations	execution time (sec)
none	4043676	1	46
eliminate impossible pairs	869700	1	14
reduce relation range	489996	2	10
pruning label sets	419184	1	5
3 schemes combined	43832	1	1.7

Table 7.1: Performance figures for the schemes for reducing the number of coefficients

7.5 Hierarchical matching

In some applications, the quality of the measurement information for some objects may be superior to that for others. For example, if line segments are being matched, longer segments are likely to have a better-defined orientation than shorter ones, and are less likely to have been spuriously generated. With the hierarchical approach, all the features shorter than some threshold length are discarded, leaving a subset of the "better" (*i.e.* longer) scene features that is nevertheless still large enough to reliably establish an overall match with the model. Given that the scene segments are more likely to have been accidentally broken than those of the model, it may also be possible to prune out the shorter model features in a similar fashion. Indeed, on this basis, each separate label set Ω_i can be pruned using a threshold length based on the length of the particular object d_i.

A preliminary match is then determined using the reduced object and label sets, and the results of this match used to determine a more accurate transformation between scene and model. Using this information it is often then possible to generate one or more additional attributes, thereby using the method of Section 7.3 to eliminate many more of the possible labellings; the matching algorithm is then rerun using the original complete node sets.

(a) image (b) map

Figure 7.3: Image of road network and map showing the respective node sets

We illustrate the technique using the road-matching example again. This time, all of segments were included: the scene contained 39 nodes and the model 303 nodes. This would have generated about 1.4×10^8 compatibility coefficients, although this number could be effectively reduced by about an order of magnitude using the method of the previous section. However the problem was still too large to run on any of the machines available to us at the time; had such a machine been available, extrapolation from other results indicated that the execution time would have been of the order of 2 minutes on a Sun Sparc10 c.p.u.

The relative position of scene and model were unknown, so no attributes were available. We used the segment length as a means of ordering the node sets; all of the nodes below a certain length were then eliminated from consideration for the first pass. This threshold was set heuristically, to generate sets of reasonable size. In the example shown in Fig. 7.3, a threshold of 15 pixels was used, yielding sets of sizes 6 and 72 nodes for the scene and model respectively. This generated about 3.7×10^4 coefficients for the first pass, converged in two iterations and executed in 1.4s on the Sun Sparc10. The result of this pass is shown in Fig. 7.4: the crosses and arrows in the image and map indicate their respective relative position and orientation, and matched and unmatched nodes are shown respectively as coarser and finer line segments. The original roads (before segmentation) are indicated on the map by the finest lines.

As Fig. 7.4 indicates, from the first pass it was possible to estimate the position and orientation of the scene with respect to the model; having done this, it was then possible to estimate the accuracy of the match. So for the second pass, all possible labellings whose estimated positions were in error by more than three standard deviations were discarded. A less rigor-

(a) image (b) map

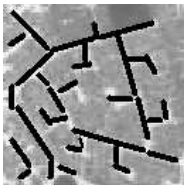
Figure 7.4: Correspondences after the 1st pass, showing the computed relative position and orientation

ous pruning was also applied to labellings with orientation mismatches: all labellings whose estimated orientations were in error by more than 30° were discarded. The more generous tolerance was adopted because the extra line segments included in the second pass were the shorter ones, and hence were likely to have larger orientation errors than those used in the first pass. In the second pass 2.2×10^3 coefficients were generated, one iteration was sufficient for convergence and the execution time was 1.0s. The results are shown in Fig. 7.5; here the relative orientation derived from the first pass was used to rotate the map to align it with the image. One short road segment (shown as a finer line than the correctly matched segments in Fig. 7.5(a)) was incorrectly matched to the null model node.

In order to demonstrate the importance of selecting the node subsets according to their saliency, the first pass was rerun, but this time nodes were selected at random to create sets for both image and map of about the same size as for the first pass above. In this case, although the match that resulted appears to be in roughly the same region of the map as before, it was in fact entirely incorrect (Fig. 7.6). This is apparent from the different direction of the dotted pointers in Figs. 7.4(b) and 7.6(b).

7.6 Parallel processing

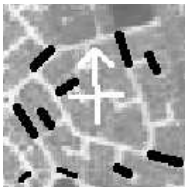
If we examine once again the labelling rule at the beginning of this chapter (p. 42), it is apparent that many of the computations can be performed independently from each other. Working from the innermost loop outwards, we can make the following observations:



(a) image

(b) map

Figure 7.5: Correspondences after the 2nd pass, using the results of the 1st pass to rotate the map



(a) image

(b) map

Figure 7.6: Incorrect result after the 1st pass, using random node selection

The summation loop of (7.1): The terms of the summation are all independent, and can therefore be computed in parallel; the summation can also be partly parallelised, but will need $\log_2 M_j$ serial operations (M_j is the size of the label set Ω_j).

The product loop of (7.1): The factors of the product are similarly all independent; the product needs $\log_2 N$ serial operations, where N is the object set size.

The loop over the label set Ω_i: The labellings within this loop are all computed independently.

The loop over the object set O: Provided that the modification to the algorithm described in Section 8.1 is not used, the labellings within this loop are all independent and can be performed in parallel. If the modification is used, the labelling probabilities calculated for object O_i are needed for the labelling of object O_{i+1}, so that parallel computation at this level is no longer possible.

The relaxation loop: Clearly each iteration of this outermost loop is dependent on the results of the previous iteration; no parallelisation is possible.

In each case where parallelisation is possible, each compatibility coefficient is only used on a single processor, and so the coefficient calculations should be performed on that processor. In the last two cases (parallelising the label or object set loops) the amount of interprocessor communication required is very small, so that a speedup linear with the number of processors can reasonably be expected.

Parallelising the algorithm at the finest level (*i.e.* in the innermost loop) is appropriate for single-instruction multiple-datastream architectures (*e.g.* the AMT DAP range) and vectorising architectures (*e.g.* the earlier Cray machines). Parallelising at the coarser levels is appropriate for more loosely-coupled machines such as the Meiko Computing Surface or even for a set of networked machines.

Chapter 8

Variations on the matching algorithm

As the details of the matching algorithm unfolded in Chapters 2 to 6, there were inevitably some heuristic decisions that had to be made. As is usual with such decisions, things do not necessarily have be done quite the way they are done. In this chapter, we discuss how changing some of these rules could lead to some slight modifications to the algorithm.

8.1 Asynchronous updating

In Chapter 4, we observed that the update rule might usefully be iterated, leading to the iterative rule repeated here:

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i|\mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j''|\mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \quad (8.1)$$

$$\Pr^{(n+1)}\left(\mathcal{L}_i^{\alpha}\right) = \frac{\Pr^{(n)}\left(\mathcal{L}_i^{\alpha}\right) Q^{(n)}\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr^{(n)}\left(\mathcal{L}_i^{\omega_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\omega_i}\right)} \quad (8.2)$$

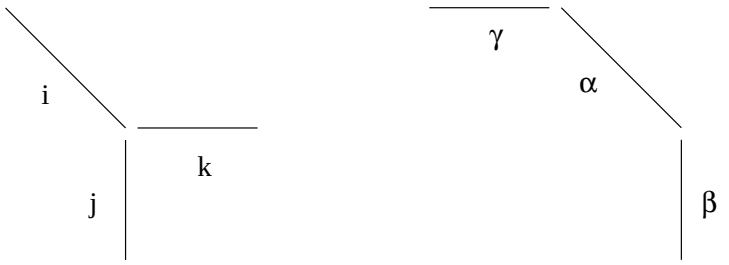
In the discussion about why the update rule might be iterated (Section 4.2), the point was made that the best possible available information concerning the labellings should be used for the prior probabilities $\Pr^{(n)}(\mathcal{L}_j^{\omega_j})$. At the beginning of an iteration, the best information available is given by the posterior probabilities calculated in the previous iteration, so that the posterior probabilities from iteration n were used as the prior probabilities for iteration $n+1$.

We can extend this argument, so that the latest available probabilities are used each time the probabilities for a new object are updated. For example in iteration n, when the the new label probabilities for object O_i are being calculated, those for objects $O_j, \forall j < i$, have already been updated. In this case, the newly-calculated values for O_j, $\Pr^{(n+1)}(\mathcal{L}_j^{\omega_j})$, can be used in place of $\Pr^{(n)}(\mathcal{L}_j^{\omega_j})$ when calculating the support (8.1) for object O_i. (However, while the probabilities for O_i are in the process of being calculated, we should of course take care that the new label probabilities for O_i itself are *not* used in (8.2); otherwise the requirement that the label probabilities should sum to unity will be violated.)

To get the most benefit from this technique, the objects (scene features) should if possible be ordered in descending order of their saliency, or "quality". In this way the best features, which should have better-defined measurements, are processed first. For example, if the features are line segments, the variances of the relation distributions for longer segments are smaller than those for shorter ones (Section 6.2.2); also they are less likely to be spurious features generated by noise (*c.f.* Section 7.5). They would therefore be ordered by length, O_1 being the longest and O_N the shortest. The use of ordered object sets is particularly useful if the update rule is not iterated: more information is available when processing the objects for which the measurement information is weaker.

8.2 Improving the contextual information for undirected line segments

When the selection of attribute and relation types appropriate to line segments were discussed in Section 6.2, this selection was essentially made using some basic principles together with some common sense. Reference was made to the distinction between directed and undirected segments (p. 38); in particular it was pointed out that if undirected segments are used as opposed to directed segments, a significant amount of information was lost due to the 180° ambiguity of orientation of the segments. This ambiguity can lead to an incorrect match, as we may see by considering the simple labelling problem shown below:



(a) scene (b) model

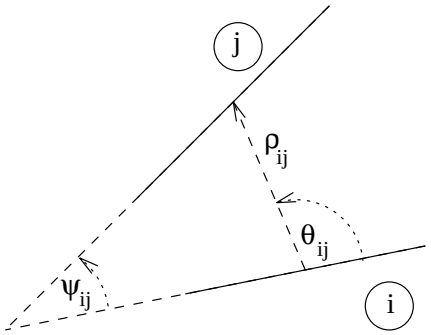
Figure 8.1: Anomalous match due to orientation uncertainty

Here, when object i is labelled with label α, it gets support (assuming one orientation) from the labelling of object j with label β. It also gets support (assuming the opposite orientation) from the labelling of object k with label γ. When we are calculating the overall support for the label α, then because we accumulate evidence of relations of object i with neighbouring nodes *individually*, this label gets strong support. However it is obvious in this case *from the whole context* that the label is wrong, and we need somehow to exploit this contextual information.

One way of incorporating this information is as follows. For each labelling, we assume that each of the scene and model segments being matched does actually have a specific orientation. There are then in effect two possible labellings to consider for each pair of segments:

one for when the segments are pointing in the same direction and one when they are pointing in opposite directions. We then calculate the support function twice, once for each of the two possible labellings, now performing calculations on angle measurements modulo 360° instead of modulo 180°. The larger value of the support function is then assumed to be the correct one.

Because each pairing of scene and model segments now has two labellings, the amount of computation could increase fourfold: a separate support function for each labelling has to be computed, and there are twice as many relations to consider. However, this extra computational burden can be reduced by a factor of two. Consider again the set of relations between pairs of line segments that was discussed in Chapter 6:



Of the three relation types, this process clearly only affects the angle relation types θ_{ij} and ψ_{ij}. Furthermore we are really only interested in the relative orientations of the particular scene and model features whose support function is currently being calculated ($\{O_i, \alpha\}$ in Fig. 8.1), not of the other features that are being used in the calculation ($\{O_j, \beta\}$, $\{O_k, \gamma\}$ etc.). In the case of the relative direction relations θ_{ij} and $\theta_{\alpha\beta}$, only the orientation of label α with respect to object O_i affects the relation values, and not the orientation of β with respect to O_j, whereas in the case of the relative orientation measurements ψ_{ij} and $\psi_{\alpha\beta}$, both relative orientations are relevant. Therefore if we continue to measure ψ_{ij} modulo 180°, but measure θ_{ij} modulo 360°, only the relative orientations of O_i and α affect the computation and not the relative orientations of O_j and β. In other words, when we are calculating the support for the labelling $\mathcal{L}_i^{\alpha}$, the ambiguities of the orientations of the features involved in other labellings are no longer relevant. Thus although we still need to calculate the support for $\mathcal{L}_i^{\alpha}$ twice, the two possible relative orientations for the labelling of the other objects do not have to be included in the computation.

This variant is again particularly appropriate if the update rule is not iterated. In the iterated case, the extra contextual information provided by this scheme is effectively provided by subsequent iterations.

8.3 An alternative factorisation of the total probability expression

In the derivation of the update rule in Chapter 3, the total probability term in (3.4) can be factored in different ways. The factorisation that leads to the preferred update rule was shown in (3.5). We describe here an alternative derivation of the rule using a different factorisation, based on an early version of the method [38].

Assumption 8.1. *The attributes and relations are derived from different measurements (*i.e. *there are no attributes of type* $\mathbf{a}'$ *— Section 3.2); hence the attributes and relations are independent.*

Comment: This in theory excludes the use of weak attributes for which the corresponding relations are strong, used in the example of Section 2.5, Example 3. In practice, attributes of type $\mathbf{a}'$ whose errors are dominated by uncertainty in the scene-to-model transformation can also usually be handled.

Thus in this version, no distinction is made between the two categories of attributes, $\mathbf{a}'$ and $\mathbf{a}''$. In this version of the algorithm, the total probability term in (3.4) is then factored as follows:

$$\begin{aligned} & p\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{a}, \mathbf{r}_i\right) = \\ & \qquad p\left(\mathbf{a}, \mid \mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}\right) \\ & \quad \times \quad \Pr\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{r}_i\right) \end{aligned} \tag{8.3}$$

The first term can be simplified if further assumptions are made:

Assumption 8.2. *The attributes are conditionally independent (see Assumption 3.5., p. 17).*

Hence:

$$p\left(\mathbf{a}, \mid \mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}\right) = p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} p\left(\mathbf{a}_j \mid \mathcal{L}_j^{\omega_j}\right)$$

The second term of (8.3) can be further factorised as follows:

$$\begin{aligned} & \Pr\left(\mathcal{L}_1^{\omega_1}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_i^{\alpha}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathbf{r}_i\right) = \\ & \qquad \Pr\left(\mathcal{L}_1^{\omega_1} \mid \mathcal{L}_2^{\omega_2}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathcal{L}_i^{\alpha}, \mathbf{r}_i\right) \\ & \quad \times \quad \Pr\left(\mathcal{L}_2^{\omega_2} \mid \mathcal{L}_3^{\omega_3}, \ldots, \mathcal{L}_{i-1}^{\omega_{i-1}}, \mathcal{L}_{i+1}^{\omega_{i+1}}, \ldots, \mathcal{L}_N^{\omega_N}, \mathcal{L}_i^{\alpha}, \mathbf{r}_i\right) \\ & \quad \times \quad \ldots \\ & \quad \times \quad \Pr\left(\mathcal{L}_N^{\omega_N} \mid \mathcal{L}_i^{\alpha}, \mathbf{r}_i\right) \\ & \quad \times \quad \Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{r}_i\right) \times p\left(\mathbf{r}_i\right) \end{aligned} \tag{8.4}$$

We then make a pair of further assumptions, which we regard as self-evident:

Assumption 8.3. *The probability of labelling $\mathcal{L}_i^{\alpha}$ is only dependent on labelling $\mathcal{L}_j^{\beta}$ if the relation $\mathbf{r}_{ij}$ is given; conversely, the relation $\mathbf{r}_{ij}$ only affects labelling $\mathcal{L}_i^{\alpha}$ if labelling $\mathcal{L}_j^{\beta}$ is given.*

Hence (8.4) can be written:

$$\Pr\left(\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_{i-1}^{\omega_{i-1}},\mathcal{L}_i^{\alpha},\mathcal{L}_{i+1}^{\omega_{i+1}},\ldots,\mathcal{L}_N^{\omega_N}\mid\mathbf{r}_i\right)=\Pr\left(\mathcal{L}_i^{\alpha}\right)p\left(\mathbf{r}_i\right)\prod_{\forall j\neq i}\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathcal{L}_i^{\alpha},\mathbf{r}_{ij}\right)$$

and (8.3) becomes:

$$\begin{aligned}
&p\left(\mathcal{L}_1^{\omega_1},\ldots,\mathcal{L}_{i-1}^{\omega_{i-1}},\mathcal{L}_i^{\alpha},\mathcal{L}_{i+1}^{\omega_{i+1}},\ldots,\mathcal{L}_N^{\omega_N},\mathbf{a},\mathbf{r}_i\right)=\\
&\qquad\Pr\left(\mathcal{L}_i^{\alpha}\right)p\left(\mathbf{r}_i\right)p\left(\mathbf{a}_i\mid\mathcal{L}_i^{\alpha}\right)\prod_{\forall j\neq i}\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathcal{L}_i^{\alpha},\mathbf{r}_{ij}\right)p\left(\mathbf{a}_j\mid\mathcal{L}_j^{\omega_j}\right)\\
&=\quad\Pr\left(\mathcal{L}_i^{\alpha}\mid\mathbf{a}_i\right)p\left(\mathbf{a}_i\right)p\left(\mathbf{r}_i\right)\prod_{\forall j\neq i}\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathcal{L}_i^{\alpha},\mathbf{r}_{ij}\right)\frac{\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathbf{a}_j\right)p\left(\mathbf{a}_j\right)}{\Pr\left(\mathcal{L}_j^{\omega_j}\right)}
\end{aligned}\tag{8.5}$$

Using this expression in (3.4) and then factorising gives:

$$\begin{aligned}
Q\left(\mathcal{L}_i^{\alpha}\right)&=\prod_{\forall j\neq i}\sum_{\omega_j\in\Omega_j}\frac{\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathcal{L}_i^{\alpha},\mathbf{r}_{ij}\right)}{\Pr\left(\mathcal{L}_j^{\omega_j}\right)}\Pr\left(\mathcal{L}_j^{\omega_j}\mid\mathbf{a}_j\right)\\
\Pr\left(\mathcal{L}_i^{\alpha}\mid\mathbf{a},\mathbf{r}_i\right)&=\frac{\Pr\left(\mathcal{L}_i^{\alpha}\mid\mathbf{a}\right)Q\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i\in\Omega_i}\Pr\left(\mathcal{L}_i^{\omega_i}\mid\mathbf{a}\right)Q\left(\mathcal{L}_i^{\omega_i}\right)}
\end{aligned}\tag{8.6}$$

The compatibility coefficient, $c(\mathcal{L}_j^{\beta},\mathcal{L}_i^{\alpha})=\Pr(\mathcal{L}_j^{\beta}|\mathcal{L}_i^{\alpha},\mathbf{r}_{ij})/\Pr(\mathcal{L}_j^{\beta})$, can be further expanded and simplified (using Assumption 8.3. again) in order to express it in terms of known quantities:

$$\begin{aligned}
c\left(\mathcal{L}_j^{\beta},\mathcal{L}_i^{\alpha}\right)&=\frac{\Pr\left(\mathcal{L}_j^{\beta}\mid\mathcal{L}_i^{\alpha},\mathbf{r}_{ij}\right)}{\Pr\left(\mathcal{L}_j^{\beta}\right)}\\
&=\frac{p\left(\mathbf{r}_{ij}\mid\mathcal{L}_j^{\beta},\mathcal{L}_i^{\alpha}\right)}{p\left(\mathbf{r}_{ij}\mid\mathcal{L}_i^{\alpha}\right)}\\
&=\frac{p\left(\mathbf{r}_{ij}\mid\mathcal{L}_j^{\beta},\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_j\in\Omega_j}p\left(\mathbf{r}_{ij}\mid\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\omega_j}\right)\Pr\left(\mathcal{L}_j^{\omega_j}\right)}
\end{aligned}$$

The form of the compatibility coefficient depends on which of the labels, if any, are null. We assume that the non-null relation p.d.f. is unimodal. If for example it were Gaussian, then

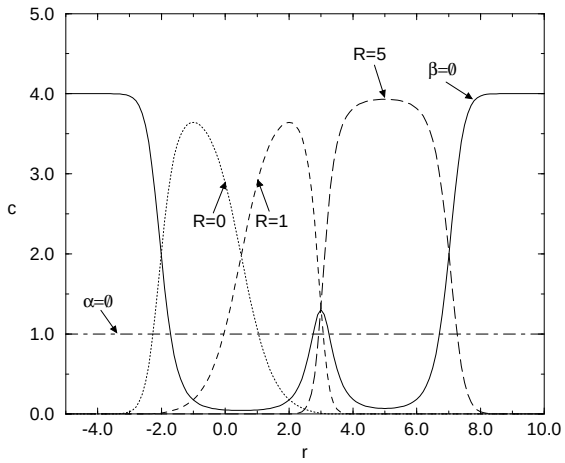


Figure 8.2: Coefficient values as a function of the scene relation

(using the notation of Section 5.3) it would have the form:

$$c\left(\mathcal{L}_j^{\beta}, \mathcal{L}_i^{\alpha}\right) = \begin{cases} \dfrac{\mathcal{N}_{\mathbf{r}_{ij}}\left(\mathbf{R}_{\alpha\beta},\, \Sigma\right)}{\frac{1-\zeta}{M}\sum_{\omega_j \in \Omega_j} \mathcal{N}_{\mathbf{r}_{ij}}\left(\mathbf{R}_{\alpha\omega_j},\, \Sigma\right) + \frac{\zeta}{D_{\mathbf{r}}}} & \text{if } \alpha \neq \emptyset,\ \beta \neq \emptyset \\ \dfrac{\frac{1}{D_{\mathbf{r}}}}{\frac{1-\zeta}{M}\sum_{\omega_j \in \Omega_j} \mathcal{N}_{\mathbf{r}_{ij}}\left(\mathbf{R}_{\alpha\omega_j},\, \Sigma\right) + \frac{\zeta}{D_{\mathbf{r}}}} & \text{if } \alpha \neq \emptyset,\ \beta = \emptyset \\ 1 & \text{if } \alpha = \emptyset \end{cases}$$

Fig. 8.2 illustrates a simple example of how the compatibility coefficient might vary as a function of $\mathbf{r}_{ij}$, showing the different forms when α or β or neither is the null label. In this example there is one relation type and three non-null model nodes (*i.e.* $M = 3$). When the model node α is not the null node, it has a relation value of 0 with itself, and 1 and 5 with the other two non-null model nodes. The noise distribution is Gaussian, with a variance of 1/2. The remaining parameters were: $\zeta = 1/(M+1) = 1/4$, and $D_{\mathbf{r}} = 100$. We make some observations about how the compatibility coefficient varies as a function of $\mathbf{r}_{ij}$:

$\alpha \neq \emptyset, \beta \neq \emptyset$: Near the point $\mathbf{r}_{ij} = \mathbf{R}_{\alpha\beta}$, the term in the denominator for the labelling $\mathcal{L}_j^{\beta}$ tends to dominate, and so the coefficient will tend towards a maximum value which is less than $1/\Pr(\mathcal{L}_j^{\beta}) = M/(1-\zeta)$. As $\mathbf{r}_{ij}$ moves away from this region towards another model relation value, the value will drop away sharply as the neighbouring p.d.f. starts to dominate in the denominator. Fig. 8.2 also shows that, if the nearest neighbour is very close, this will push the peak value in the opposite direction

$\alpha \neq \emptyset, \beta = \emptyset$: The maximum value has $1/\zeta$ as an asymptote. The coefficient approaches a minimum value as $\mathbf{r}_{ij}$ approaches each of the $\mathbf{R}_{\alpha\omega_j}, \omega_j \neq \emptyset$.

$\alpha = \emptyset$: Unity.

Thus the compatibility coefficients formed in this way have some interesting differences from those derived in Chapter 3:

- They are by definition asymmetrical in the two labellings $\mathcal{L}_i^{\alpha}$ and $\mathcal{L}_j^{\beta}$ (from (8.6)) — they represent the ratio of the probability of the labelling $\mathcal{L}_j^{\beta}$, given its relations with another labelling $\mathcal{L}_i^{\alpha}$, to that without. In other words, they can be viewed as a kind of elemental support function, indicating the support given by labelling $\mathcal{L}_i^{\alpha}$ to the labelling $\mathcal{L}_j^{\beta}$.
- They are intrinsically normalised, and thus dimensionless.
- If the supporting label is null ($\alpha = \emptyset$), this provides support neither for nor against the labelling $\mathcal{L}_j^{\beta}$, regardless of the label β. This "no view" value is unity.
- The converse is not true: if the labelling being considered is the null labelling ($\beta = \emptyset$), but the supporting one is not, the supporting labelling *does* provide information. If the scene relation $\mathbf{r}_{ij}$ is consistent with the null label being in the vicinity of one of the non-null labels, the coefficient has a low value, and vice versa. Thus the support for the null label will be higher if none of the other labels is nearby.

8.4 Excluding the attributes from the iterations

The relaxation rule (4.1, 4.2) derived in Chapter 4 incorporates both attributes and relations at each stage in the iterated rule. This differs from the earlier version in [13], which incorporated only the relations at each iteration. We see no particular merit in this earlier version, but we include it here for completeness.

The rule was derived as follows. As in the previous section, we assume that there are no attributes of type $\mathbf{a}'$ (Section 3.2). Bayes rule is applied to the update rule of (3.10, 3.11) in order to replace the prior probabilities by probabilities conditional on the attributes:

$$\begin{aligned} Q(\mathcal{L}_i^{\alpha}) &= \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) \Pr\left(\mathcal{L}_j^{\omega_j} \mid \mathbf{a}_j\right) \\ \Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}, \mathbf{r}_i\right) &= \frac{\Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i\right) Q\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr\left(\mathcal{L}_i^{\omega_i} \mid \mathbf{a}_i\right) Q\left(\mathcal{L}_i^{\omega_i}\right)} \end{aligned}$$

This rule thus describes how to add relational information only to a set of probabilities that already contain information of the attributes, and thus can be used as an iterative rule of the form:

$$\begin{aligned} Q^{(n)}(\mathcal{L}_i^{\alpha}) &= \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij} | \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \\ \Pr^{(n+1)}\left(\mathcal{L}_i^{\alpha}\right) &= \frac{\Pr^{(n)}\left(\mathcal{L}_i^{\alpha}\right) Q^{(n)}\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr^{(n)}\left(\mathcal{L}_i^{\omega_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\omega_i}\right)} \end{aligned}$$

The probabilities are initialised by applying Bayes rule once more:

$$\Pr^{(0)}\left(\mathcal{L}_i^{\alpha}\right) \equiv \Pr\left(\mathcal{L}_i^{\alpha} \mid \mathbf{a}_i\right) \propto p\left(\mathbf{a}_i \mid \mathcal{L}_i^{\alpha}\right) \Pr\left(\mathcal{L}_i^{\alpha}\right)$$

Chapter 9

Evaluating the performance of the matching algorithm

We fist consider some ways in which the performance of the matching algorithm might be assessed. We then discuss examples of three different applications of the algorithm. The first is the road matching example used earlier (Chapters 1, 6, 7). This application is used to demonstrate the convergence rate of the algorithm, and also to test its sensitivity to the parameter values and to scaling errors. The second is a stereo matching example, and the third is an application to establish the correspondence between an image and a 2-D projection of a 3-D model.

9.1 Evaluation of the match accuracy

When using the algorithm as a component of a larger machine vision application, it is important to be able to estimate the quality of the result in some way. There are three useful heuristic indicators: the distance between corresponding features once the scene-to-model transformation has been determined, the amount of support for the MAP labellings, and the number of null matches.

Distance between corresponding segments

Once the MAP labellings have been determined, the scene-to-model transformation is determined by performing a least-mean-square fit between the corresponding features. From this calculation it is also possible to find a measure of the average distance between matched pairs of features. At present a heuristic approach is taken, using just the centre of gravity of the features.

Labelling support

Let us look again at the support function (4.1) for the labelling $\mathcal{L}_i^{\alpha}$ defined in Section 4.3:

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i|\mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j''|\mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right)$$

On convergence (*i.e.* as $n \to \infty$), each object O_j will have one label, the MAP label, for which the probability $\Pr^{(\infty)}(\mathcal{L}_j^{\omega_j^{MAP}})$ should equal unity, and the remaining labelling probabilities should have a value of zero. Thus there is just one non-zero term in the summation in the above expression, which can be rewritten:

$$Q^{(\infty)}(\mathcal{L}_i^{\omega_i^{MAP}}) = p\left(\mathbf{a}_i|\mathcal{L}_i^{\omega_i^{MAP}}\right) \prod_{\forall j \neq i} p\left(\mathbf{r}_{ij}|\mathcal{L}_i^{\omega_i^{MAP}}, \mathcal{L}_j^{\omega_j^{MAP}}\right) p\left(\mathbf{a}_j''|\mathcal{L}_j^{\omega_j^{MAP}}\right)$$

We assume that each of the (non-null) distributions involved is unimodal, with maximum corresponding to an exact match. The maximum possible value of this MAP support function can readily be calculated. The ratio of the actual value to this maximum, averaged over all the scene nodes, gives a measure of the goodness of fit of the overall result.

Number of null matches

Where good labellings cannot be found, the algorithm usually assigns the null label. Thus an abnormally large number of null labels usually indicates some problem with the overall match. The drawback with this indicator is that experience is required in the particular application in order to be able to judge how many null matches are acceptable.

9.2 Road matching

We tested the algorithm on a series of 19 aerial images (Fig. 9.1) that were extracted from a larger image, which mostly consisted of a suburban road network. A relaxation algorithm that detected intensity ridges (based on [29]) was used to locate the roads, to which line segments were fitted using a least-mean-square fit algorithm. These line segments are clearly undirected (in the sense of Section 6.2.5). All of the images corresponded to locations on the same map, reproduced in Fig. 9.2. The image size was chosen in order to give the feel of a typical navigation problem: the image is significantly smaller than the map, and the task is to find the position and orientation of the image with respect to the map. The orientation of the images was not assumed to be given, while the image and model scales are closely matched; the unknown transformation can therefore be assumed to be Euclidean, with no attributes and three relation types (Section 6.2.2). All image and map segments shorter than 8 pixels were discarded. The noise model used the covariance matrix method of Section 6.2.2.

The correct position and orientation of the image with respect to the map are known in these tests, so the match accuracy can be found from the mean position and orientation of the matched segments. The algorithm generated consistent matches for all 19 images, and all were correct except for image 17. Unfortunately in this case an incorrect, but highly plausible, match was found instead of the correct one (Fig. 9.3). In the figure, the incorrect labels are shown as dark segments, and the correct position of the image is also indicated.

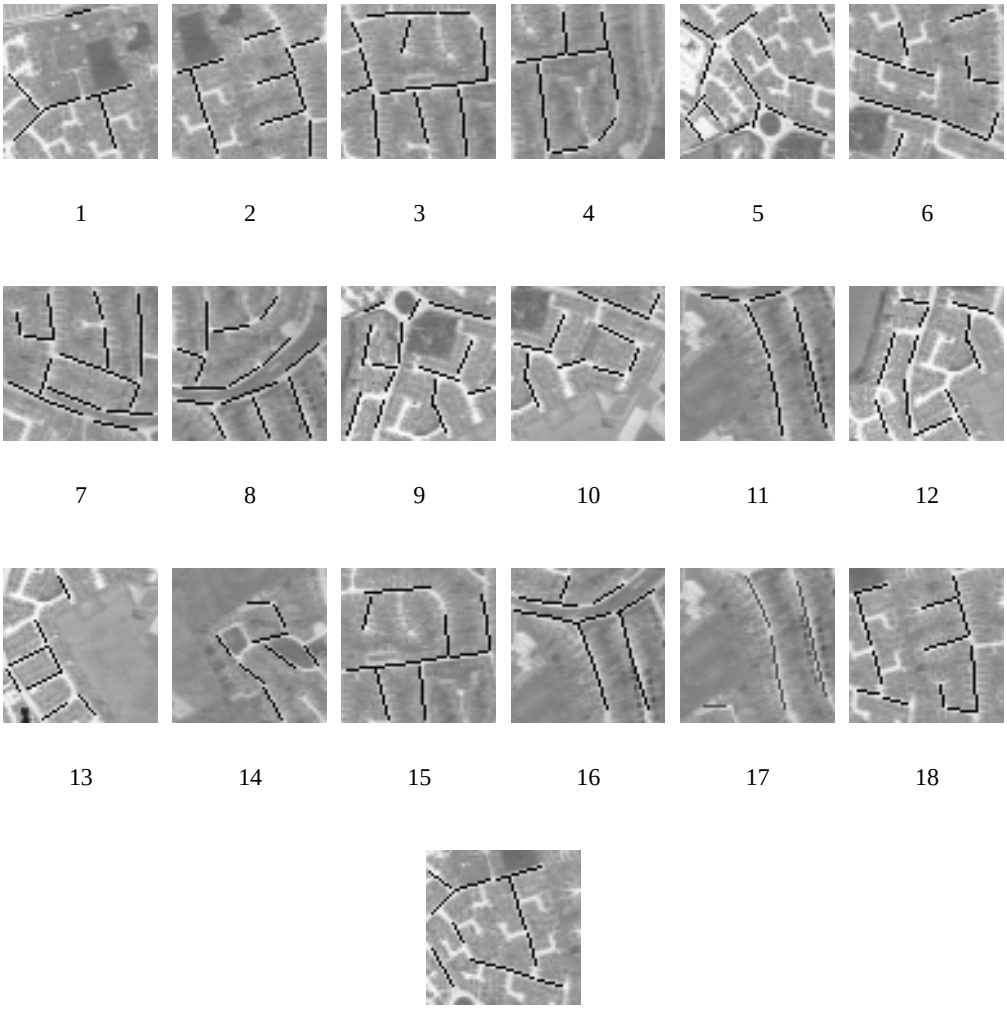
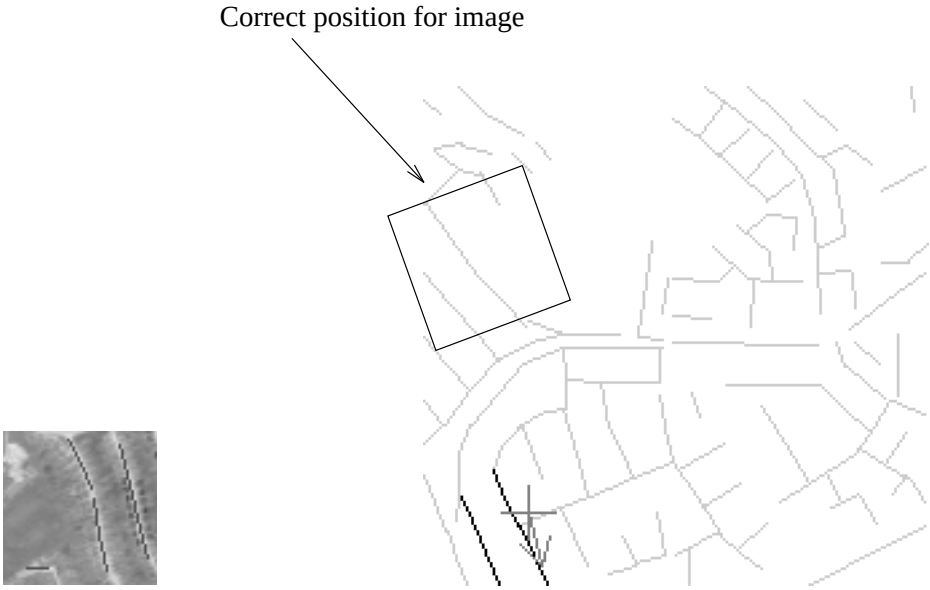


Figure 9.1: The 19 test images, with line segments superimposed

Figure 9.2: The map



(a) image 17

(b) Map, showing incorrectly-labelled segments

Figure 9.3: Map showing incorrect labelling for image 17

9.2.1 Convergence rate

The number of iterations needed are shown in Table 9.1. A good match is defined as one for

Fig.	1	2	3	4	5	6	7	8	9	10
iterations for good result	1	1	1	2	32	2	3	2	3	2
iterations for final result	2	2	3	2	33	4	4	3	5	3

Fig.	11	12	13	14	15	16	17	18	19
iterations for good result	2	2	1	3	2	2	2	2	2
iterations for final result	3	11	3	4	2	2	3	3	2

Table 9.1: Number of iterations required for convergence

which all the labels are basically "correct", except that perhaps one or at most two are incorrectly matched to the null node. The algorithm was terminated when each MAP labelling probability had a value of at least 0.9999. When the final labelling is reached, typically the null labels are replaced by the correct ones; also a non-null labelling may change in cases where there is a choice of good labels. The result for image 5 deserves further comment. There were some spurious scene nodes and missing model nodes; also some of the scene nodes were poorly orientated with respect to their correct label segments. All of the segments were labelled as null until the 32nd iteration.

9.2.2 Sensitivity to parameter values

In this application four parameter values are needed: the null match p.d.f. value, the prior probability of a null label and the two variances associated with the position of a segment endpoint. A rationale for the calculation of default values for each one has already been discussed; however it is not obvious that the values generated will always be very precise. We accordingly examine the sensitivity of the labelling result to each of them in turn. In each case, the default values are used for the remaining parameters. We used image 9 for the tests; for this image the default values of the parameters generated one (genuinely) null label, corresponding to a missing model feature.

Null p.d.f. value

Since there are no attributes in this instance, we are concerned only with the value of the relation null p.d.f. This is given by $1/D_{\mathbf{r}}$, where $D_{\mathbf{r}}$ is the size of the relation space (Section 5.2.3). There are two angle relations, θ and ψ, and one distance relation ρ (Section 6.2.2). The modification of Section 8.2 is used, so θ has a range of 360°, while ψ has a range of 180°. The image is square, with side 60 pixels, so ρ has a range of 60 pixels. Hence $D_{\mathbf{r}} = 360 \times 180 \times 60$, and its reciprocal gives the default p.d.f. value. This is the default value used to generate the results in Table 9.1.

We would expect the null p.d.f. value primarily to affect the number of null labellings. In order to test the sensitivity of the algorithm to this value, we multiplied the default value

multiplier for null p.d.f. value	0.001	0.01	0.1	0.5	1	2	≥ 5
no. of null matches	1	1	2	2	2	2	all
no. of iterations for final result	2	2	4	4	5	9	1

Table 9.2: Variation of labelling with $D_{\mathbf{r}}$

by a range of factors, the results being shown in Table 9.2. The results suggest that the default p.d.f. value is on the borderline of the range of good values, and that a lower value would be safer. The value used as the default was based on the scene parameters; using the model parameters would give a lower value for the p.d.f. Otherwise the algorithm is very insensitive to this parameter.

Null label prior probability ζ

The default value for this parameter (Section 5.1) is $1/(1+M)$, where M is the number of non-null labels; in this example, $M = 161$. We used the same image (image 9) for this test, so that the default value of ζ generated one null label, as before. Again, we would expect the value of ζ primarily to affect the number of null labellings. The results were surprising: values of ζ up to 0.95 gave identical results to the default value of 1/162, with two null labels. Setting ζ to zero forced all labels to be non-null — inevitably, because from the update rule (4.2), once a labelling probability has a zero value, it will remain at zero. A nearby (but incorrect) segment was selected instead. Similarly, setting ζ to one forced null labels throughout.

Variances of the segment endpoint position, σ_{xx} and σ_{yy}

When calculating the default value for σ_{xx}, the errors of the segment endpoints along the segment, Δa_x and Δb_x, were assumed to be uniformly distributed over the range $\pm 1/2l$ (see Section 6.3). A range of values either side of this default value was then tested (Table 9.3). In all cases, the non-null matches were good ones; in the case for which there were no null

multiplier for σ_{xx}	≤ 0.03	0.1	0.3	1	3	10	30	100	≥ 300
no. of null matches	all	6	4	2	1	1	1	0	all
iterations for final result	1	10	6	5	3	6	6	141	1

Table 9.3: Variation of labelling with σ_{xx}

matches (multiplier = 100), the scene node that should have been labelled with the null node was labelled with a plausible non-null model node. It appears from the table that the default value is roughly in the middle of the usable range of values, and that the actual value chosen is not very critical.

A subjective examination of image 9 suggested that a reasonable value for σ_{yy} would be in the region of 1 pixel^2. Values either side of this were tried, and the results shown in Table 9.4. Again, the algorithm seems to be tolerant to a reasonably wide range of values.

σ_{yy}	≤ 0.02	0.05	0.1	0.2	0.5	1	2	5	10
no. of null matches	all	9	9	5	3	2	2	1	all
iterations for final result	1	16	4	6	7	5	5	4	1

Table 9.4: Variation of labelling with σ_{yy}

Some general comments are in order concerning the variance values, which are illustrated by the above results. For both variances, using too small a value restricts too severely the permitted degree of misalignment between scene and model; this causes null labels to be chosen, as might be expected. When large values are chosen, the non-null p.d.f.s become broader, with a lower peak value. There are two consequences of this:

- because the p.d.f.s are widened, they lose their discriminatory power, which may cause slow convergence, and
- once the peak values of the p.d.f.s are lower than the value of the null p.d.f., null labels will necessarily be chosen.

9.2.3 Sensitivity to scaling errors

Here we examine the sensitivity of the method to scaling errors, and test the effectiveness of modelling the scaling error as noise as described in Section 6.2.3. For this experiment, the map was misscaled by a range of factors. Segments that were shorter than 8 pixels in the *scaled* map were discarded. Default values for all of the parameters were used. The results are shown in Table 9.5, indicating the performance of the algorithm both with and without

scale		≤ 0.4	0.5	0.7	0.8	1.0	1.1	1.2	1.3	1.4	1.5
good match?	with extra term	no	yes	yes	yes	yes	yes	yes	yes	yes	null
	without extra term	no	no	no	yes	yes	no	no	null	null	null
iterations for final result	with extra term	5	7	4	3	2	3	3	14	43	1
	without extra term	13	7	4	4	2	20	21	1	1	1

Table 9.5: Variation of labelling with scaling error

the extra term in the relation distance variance. Note that, if the map scale factor is s, a value of $|1-s|$ is used for the factor ε of the extra variance term. These results indicate that, by incorporating the extra term for the unknown scale error in the distance error variance, the range of unknown scale factors that the algorithm can tolerate is significantly increased. As ε is increased, the results indicate that more iterations are needed. This is to be expected, since the effect is to degrade the more distant relations, which means that more iterations will be needed to spread the contextual information across the whole scene (*c.f.* the first termination condition in the list in Section 4.3, p. 21).

9.3 Stereo image matching

For this application we used a stereo pair of images of an office scene, each image measuring 246 by 246 pixels. The edges in the two images were detected and fitted with directed

straight line segments. We assumed that the two images have the same orientation, so that the segment orientation was used as an attribute, with a standard deviation of 5°; there were therefore two relation types (Section 6.2.3 — "Unknown translation"). We were not able to exploit the epipolar constraint with the method as it stands; instead we assumed that the features in one image were within 40 pixels horizontally and 20 pixel vertically of the features in the other image; this information was used to prune the labelling sets (Section 7.3). Line segments less than 6 pixels long were discarded. The left-hand image, which contained 229 segments, was used as the model and the right-hand one, with 238 segments, as the scene. We increased the prior probability of a null labelling ζ to 0.25 to reflect the greater likelihood of null matches.

Fig. 9.4 shows a typical result of the matching of the stereo image pair. Note that the algorithm permits more than one scene node (the right-hand image in this case) to be matched to the same model (left-hand image) node. Here corresponding line segments in the two images are indicated by white lines; black lines correspond to those that remained unmatched. We

(a) (b)

Figure 9.4: Matching edge segments from a stereo pair of images

had no ground truth for this problem, but we were able to identify only one or two incorrectly-matched segments. In particular it is worth noting that the segments that constitute the extra window panes in the right-hand image are not matched (which is correct). However the extra picture that appears on the wall in the left-hand image is (in part) incorrectly matched to the middle of the three pictures in the right-hand image.

9.4 3-D to 2-D matching

This example was a simulation of a typical underwater inspection task in the offshore petroleum industry. The structure being inspected was a scale model of part of a typical offshore oil rig

jacket (Fig. 9.5), for which a 3-D CAD model was provided.

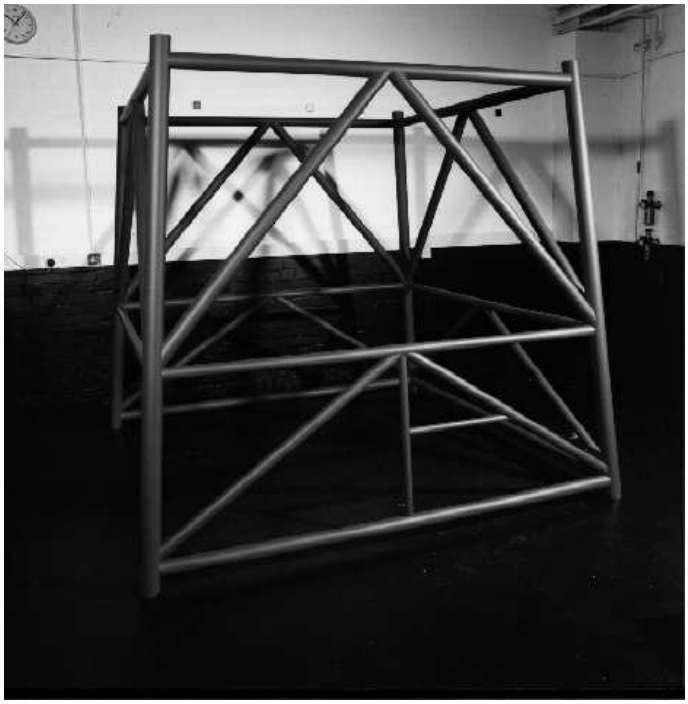


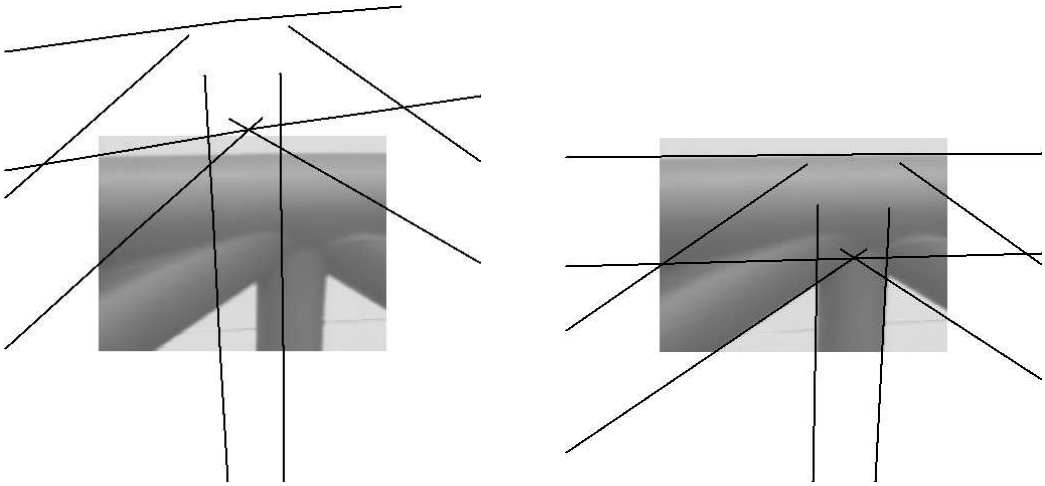
Figure 9.5: Scale model of jacket

In real life, the camera would be mounted on an underwater remotely-operated vehicle (ROV). Because the ROV is moving, both under its own power and under the influence of unknown currents, the pose (position and orientation) of the ROV with respect to the jacket is normally difficult to ascertain accurately. The task here is, by using an initial rough estimate of the ROV position, to establish the correspondence between the image and the model, and thereby to generate a more accurate estimate of the ROV position. The procedure was as follows:

1. Using the initial guess of the camera pose, project the cylinder edges from the 3-D CAD model onto the image plane as a set of directed line segments.
2. Extract the cylinder edges from the image in the form of a set of directed line segments.
3. Establish the match between the two sets of segments.
4. Using a gradient descent method, with a cost function based on the mismatch between the segment pairs, iteratively refine the camera pose.

We used the matching algorithm to implement Step 3. It is relatively easy to establish that the camera is upright, so we were again able to use the segment orientation as an attribute, with two relations as in the previous example (Section 9.3). Because corresponding segments were disparate in length, it was found to be useful to break them up into shorter segments, of length roughly corresponding to the cylinder diameter.

An example is shown in Fig. 9.6. In Fig. 9.6(a), the image is shown with a projection of the model superimposed; this projection was made using the initial estimate of the camera pose. In Fig. 9.6(b) the model is again projected onto the image, this time using the updated camera pose generated by the system.



(a) initial projection (b) final projection

Figure 9.6: Projections of the model onto the image

Chapter 10

Implementing the matching algorithm

At the time of writing, the algorithm exists in the form of a pair of C programs:

gf2arg - generates an attributed relational graph from a set of geometric features. It will optionally generate the corresponding sets of covariance matrices.

matcher - from the scene and model graph, generates a list of objects with their corresponding labels.

Other utilities are also available to assist the visualisation of the results, including:

find_where - uses a least-mean-square algorithm together with the label correspondences to fit the scene features to those of the model. Hence it finds the position and orientation of the scene w.r.t. the model.

xgf, xmgf - two utilities, each of which can create a display of geometric features optionally overlaid on a background image. Between them, they were used to create most of the figures in this thesis.

The complete labelling algorithm can be summarised by the following steps:

1. Using knowledge of the available measurement types and of the scene-to-model transformation, select the attribute and relation types appropriate to the problem (Chapter 2).
2. From the model features, calculate the model attributes and relations.
3. From the scene features, calculate the scene attributes and relations.
4. If the method of Section 6.2.2 is being used, use the scene noise model together with knowledge of the scene-to-model transformation to generate the attribute and relation distributions; otherwise estimate them.

5. Evaluate the attribute and relation p.d.f. terms, $p(\mathbf{a}_i|\mathcal{L}_i^{\alpha})$ and $p(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\beta})$, using the methods of Chapters 5 and 6. We assumed a Gaussian distribution for the non-null p.d.f.s.

6. Set iteration counter $n=0$. Initialise the probabilities $\Pr^{(0)}(\mathcal{L}_i^{\alpha})$, using the prior probabilities whose values are determined from (5.1) by the method described in Section 5.1:

$$\Pr^{(0)}(\mathcal{L}_i^{\alpha}) = \Pr(\mathcal{L}_i^{\alpha}) \tag{10.1}$$

7. For each label of each object, compute the support function (4.1):

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = p\left(\mathbf{a}_i|\mathcal{L}_i^{\alpha}\right) \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} p\left(\mathbf{r}_{ij}|\mathcal{L}_i^{\alpha},\mathcal{L}_j^{\omega_j}\right) p\left(\mathbf{a}_j''|\mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \tag{10.2}$$

8. For each label of each object, find the updated probability $P^{(n+1)}$ of the match (4.2):

$$\Pr^{(n+1)}\left(\mathcal{L}_i^{\alpha}\right) = \frac{\Pr^{(n)}\left(\mathcal{L}_i^{\alpha}\right) Q^{(n)}\left(\mathcal{L}_i^{\alpha}\right)}{\sum_{\omega_i \in \Omega_i} \Pr^{(n)}\left(\mathcal{L}_i^{\omega_i}\right) Q^{(n)}\left(\mathcal{L}_i^{\omega_i}\right)} \tag{10.3}$$

9. Using one or more of the termination criteria from Section 4.3: if, for each object O_i, the termination criteria are satisfied, go to step 10; otherwise increment the iteration counter n and go to step 7.

10. For each object O_i, choose the labelling that has the highest probability.

The flow of data through the algorithm is shown in Fig. 10.1.

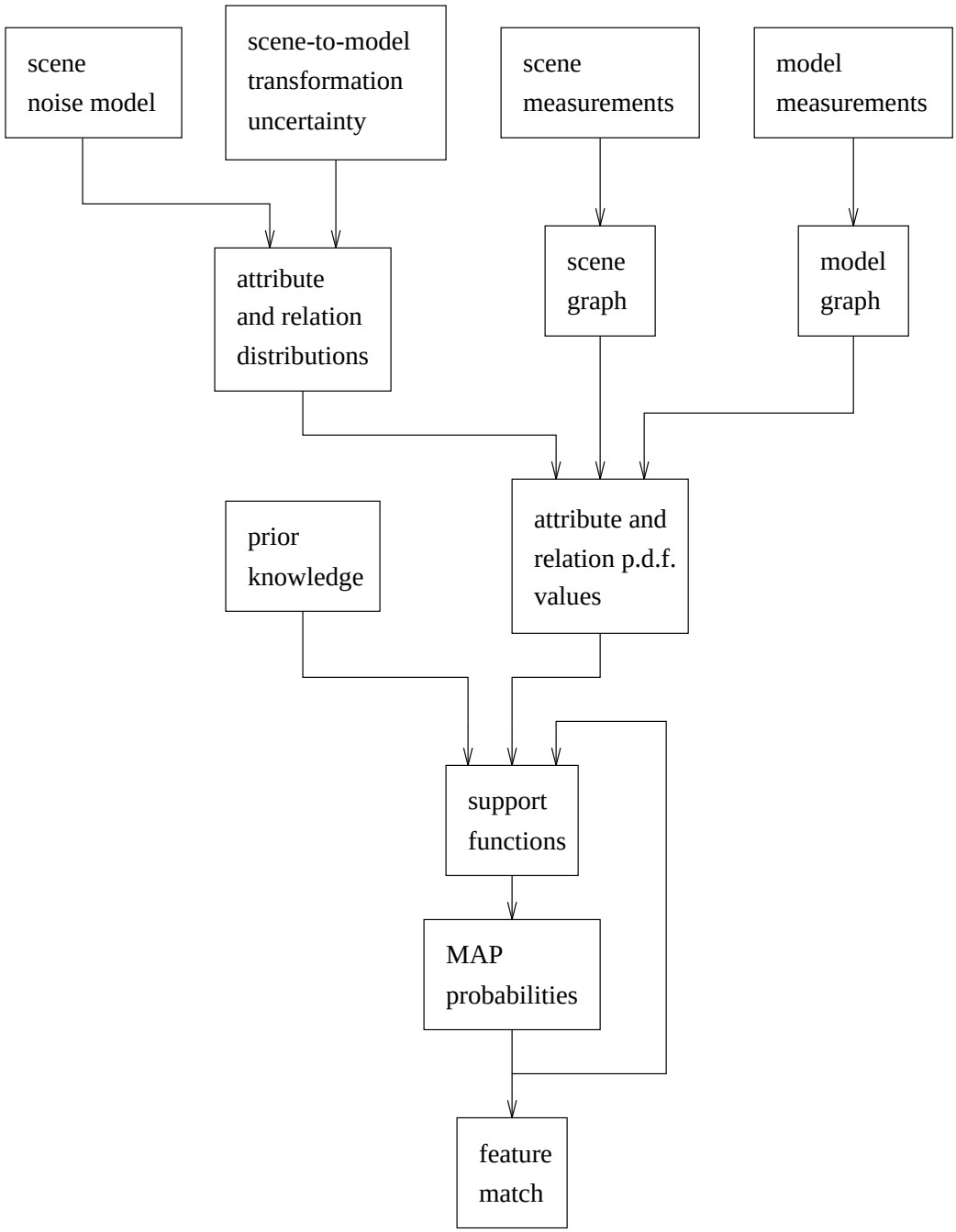


Figure 10.1: Data-flow chart

Chapter 11

A comparison of the method with other graph matching techniques

The graph matching problem is an important one in computer vision, and many workers have described a multitude of different methods over the years. We briefly discuss some of these methods, and then discuss the relationship between our method and those other methods that are more closely related to it.

11.1 A review of labelling methods

The matching problem has been approached in many different ways in the computer vision literature [2, 3, 6, 9, 10, 11, 20, 23, 24, 27, 28, 49, 52, 54, 58, 60, 63]. The early attempts, still widely popular, are based on graph search methods. These techniques generally rely on heuristic measures to reduce the complexity of the inherently NP-complete search problem to a more manageable level. More recent are the efforts based on energy minimisation using simulated annealing [1, 26, 35], mean field theory [25] or deterministic annealing [7, 8, 31, 41, 61], neural networks [53], and relaxation labelling [4, 5, 20, 22, 24, 33, 45, 46, 50, 57].

The relaxation labelling approach in particular has the advantage that it replaces the basic NP-complete search method with one of polynomial complexity. Although probabilistic relaxation has been shown to offer a very effective method for attributed relational graph matching, its foundations and consequently the relaxation process design methodology are heuristic. The work of Kittler and Hancock [40, 29], directed towards theoretical underpinning of probabilistic relaxation using a Bayesian framework, proved very successful. It led to the development of an evidence-combining formula which fuses observational and prior contextual information in a theoretically sound manner. The polynomial combinatorial complexity has been reduced even further using the concept of a label configuration dictionary. Unfortunately, the methodology is applicable only to low level matching problems such as edge or line postprocessing. The main reason for this limitation is that, after the initialisation stage, in which observations are used to compute the initial noncontextual probabilities for the object labellings, the process does not make use of the measurements.

Some workers have attempted to remedy this problem by heuristic means. Yamamoto [62] used an information-theoretic approach to derive a compatibility measure from relational measurements, and then from this measure generated by heuristic means compatibility coefficients to fit the relaxation method of Rosenfeld *et al.* [50]. Li [42] incorporated relational measurements into compatibility coefficients which figure in his probability-updating formula. In this way he overcame a major criticism of the probabilistic relaxation approach in [40], as measurements are used in all stages of the iterative process to encourage a consistent labelling. However, his solution retained the heuristic framework of probabilistic relaxation as introduced by Hummel and Zucker [33]. In particular, the compatibility and support functions are specified heuristically.

There is a similarity between the structure of the probabilistic relaxation algorithms and some neural network algorithms [30, 36, 47]. In [14] the authors explicitly draw a parallel between perceptron-based neural networks and the method described here; this parallel is summarised in Section 11.3.

In our approach, the MAP rule was developed into a method for assigning probabilities to the labellings; a heuristic decision was then made to iterate this rule to achieve a consistent result. We offer no proof that a consistent solution will be obtained, although we find that in practice it is. There have however been some attempts to place the relaxation aspect of the labelling problem on a sounder theoretical footing. For instance the method of Hummel and Zucker [33] was shown to lead to a consistent solution (albeit using a different definition of consistency). However, the form of their update rule and support function are different from ours, and in practice converges much more slowly (see [42]). Recent work by Stoddart *et al.* [55, 56] show how methods similar to ours can be cast as optimisation problems.

11.2 Comparison of this method with other relaxation methods

In this section we compare our method to those of other workers, in particular to the methods of Kittler and Hancock [40], Rosenfeld, Hummel and Zucker [50], Hummel and Zucker [33] and Li, Kittler and Petrou [43].

The derivation of our method is similar in principle to that of Kittler and Hancock, with the important difference that we include binary as well as unary information. In both cases a MAP probability is sought, and in both cases the support function is initially derived in the form of a multiple summation of a product, which is of exponential complexity (equation 3.11 in our case). In [40], this problem is resolved in one of two ways: either the model is sufficiently small that a dictionary method may be used, or it is assumed that the neighbouring nodes that interact directly with a given node are independent of each other, which enables the factorisation of the support function. With our method, the inclusion of the binary information leads to a form of the support function which can be factorised without the need for any further assumptions; this means that we can apply it to large problems in which each node interacts with all of the other nodes. In this factorised form our support function is then in a similar form to that derived in [40]. This type of product-of-sum support function has also been derived, using different approaches, by several other authors [34, 39, 44, 64].

If we compare our updating rule (4.2) with that proposed by Rosenfeld *et al.*, we can see that they are of the same form (our support function Q being related to their support function q

by $Q = 1+q$), although the form of the support function (4.1) is different. However we can show that their method is the limiting case of our method if we assume that the contextual information conveyed by the binary measurements $\mathbf{r}_{ij}$ is small, and if there are no attributes.[1] From the discussion of compatibility coefficients in Section 5.3, we may normalise the p.d.f. in (4.1) without affecting the overall result; thus we may put

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = \prod_{\forall j \neq i} \sum_{\omega_j \in \Omega_j} c\left(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \tag{11.1}$$

where

$$c\left(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right) = \frac{p\left(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}\right)}{p_0} \tag{11.2}$$

p_0 being some value of the p.d.f. for which the match $\mathcal{L}_i^{\alpha}$ is expressing support neither for nor against the match $\mathcal{L}_j^{\beta}$. The value of the uniform p.d.f. that results from a null labelling (Section 5.2.2) is an appropriate value for p_0. Thus $c(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})$ is a positive quantity, which we can view as expressing compatibility between the matches $\mathcal{L}_i^{\alpha}$ and $\mathcal{L}_j^{\beta}$. In particular if the match $\mathcal{L}_i^{\alpha}$ supports the match $\mathcal{L}_j^{\beta}$, r is greater than one, and vice versa.

If the influence of match $\mathcal{L}_i^{\alpha}$ on match $\mathcal{L}_j^{\beta}$ is small (as was assumed for example in [38]), we can put

$$c(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}) = 1 + \pi(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta}) \tag{11.3}$$

where we are assuming that the quantities $\pi(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})$ are some small numbers, *i.e.*

$$|\pi(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})| \ll 1 \tag{11.4}$$

Substituting (11.3) into (11.1), we obtain:

$$Q^{(n)}(\mathcal{L}_i^{\alpha}) = \prod_{j \in N_i} \left\{ 1 + \sum_{\omega_j \in \Omega_j} \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \pi\left(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) \right\} \tag{11.5}$$

which, on expanding the product and ignoring second and higher order terms, becomes

$$\begin{aligned} Q^{(n)}(\mathcal{L}_i^{\alpha}) &\simeq 1 + \sum_{j \in N_i} \sum_{\omega_j \in \Omega_j} \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \pi\left(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) \\ &= 1 + \sum_{j \in N_i} \sum_{\omega_j \in \Omega_j} \Pr^{(n)}\left(\mathcal{L}_j^{\omega_j}\right) \left\{ c\left(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\omega_j}\right) - 1 \right\} \end{aligned} \tag{11.6}$$

Thus if the πs in this form of the support function are equivalent to the weighted correlation coefficients of Rosenfeld *et al.*, we can see that the two methods are equivalent. In practice however, the p.d.f. $p(\mathbf{r}_{ij} \mid \mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})$ is likely to have variances that are small compared to the range of the corresponding measurements, with correspondingly large peak values and rapidly diminishing tails (section 5.2.1); therefore the assumption that $|\pi(\mathcal{L}_i^{\alpha}, \mathcal{L}_j^{\beta})| \ll$

[1] Alternatively, attributes could be incorporated in the initialising of the probabilities, as described in Section 8.4.

1 is not valid. Attempts to use the method by scaling down the πs were not satisfactory: matches were more frequently incorrect than those obtained using the support function of equation (4.1), and the number of iterations required for convergence was typically greater by about one order of magnitude.

There is another important difference from the work of Rosenfeld *et al* and especially also from that of Hummel and Zucker [33]: the compatibility coefficients used by these authors were symmetric with respect to their arguments, while those that we have derived in general are not (see Chapter 6). Thus in our method the match $\mathcal{L}_i^{\alpha}$ does not necessarily give to match $\mathcal{L}_j^{\beta}$ the same support as the latter gives to the former. Hummel and Zucker showed that the updating scheme using symmetrical coefficients is equivalent to the optimisation of a global cost function. The asymmetry of our compatibility coefficients implies that this equivalence does not apply in our case.

The method of Li *et al.* uses a similar form of support function to that of (11.6), and their compatibility coefficients π are similar, although they are derived heuristically. The updating rule is a modified form of the projected gradient algorithm of Hummel and Zucker. From the analysis earlier, it is clear that the approach of Li *et al.* corresponds to the case of low contextual information. This assumption clearly does not hold, since once again the variances of the distributions they used are small. Also their scheme needed many more iterations to converge (typically 30 – 50), whereas our scheme reaches a stable solution typically within a couple of iterations.

11.3 Comparison of this method with the neural network approach

Neural networks have often been used in labelling problems. However, from our point of view, they have two significant drawbacks: they usually require a long training period, particularly for problems with many features, and their design, in terms of the number of nodes needed and the form of activation functions, is largely heuristic. In [14] it was shown how a MAP labelling algorithm could be represented in terms of traditional neural network components. The particular form of the algorithm used for the comparison was the variation of our method described in Section 8.4, in which the probabilities are initialised with values that are conditional on the attributes. In the neural network representation, the algorithm takes the form of a perceptron-based network, and explicitly indicates the number of nodes needed and the form of the activation functions. The architecture is shown in Fig. 11.1. The evaluation of the support function in the labelling algorithm is represented as the two perceptron layers in the network. The inputs on the first iteration are the probabilities conditional on the attributes. The compatibility coefficients, which contain the scene and model relational information, form the weights in the first layer. This is similar to a traditional perceptron design, in which the model information is contained in the weights; the difference is that these weights no longer have to be learnt, but are given by the theory. The form of the activation functions is no longer heuristic; they are logarithmic and exponential for the first and second layers respectively. There is also a final auxiliary layer that provides a normalisation function to ensure that the labelling probabilities sum to unity. These probabilities are then fed back to the input layer if the iterative version of the algorithm is used.

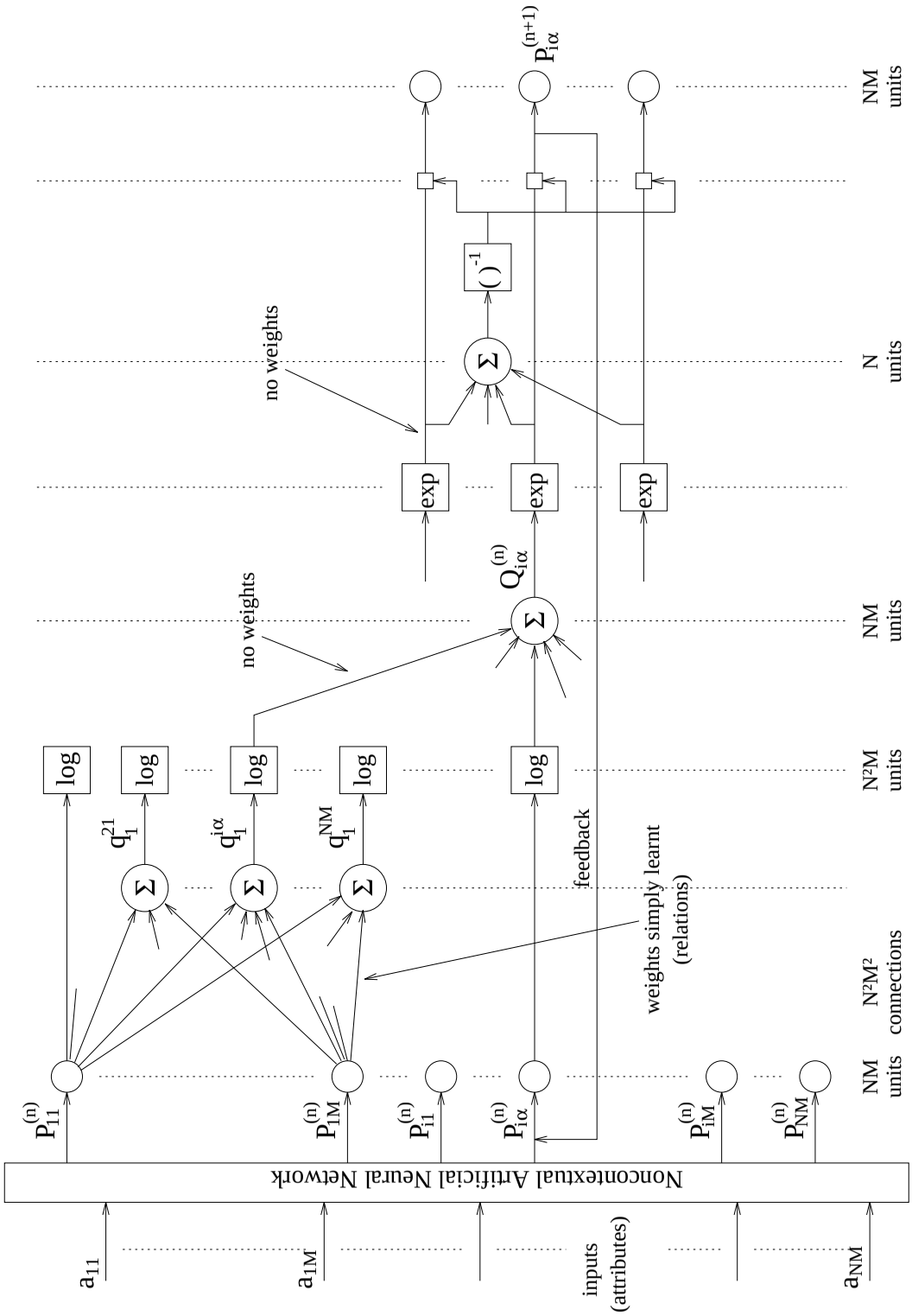


Figure 11.1: Multilayer perceptron representation of MAP labelling scheme

An alternative neural net representation was also briefly explored; this version is shown in Fig. 11.2. For simplicity, the attribute information is not included in the figure. Again two layers of perceptron-like nodes are used to calculate the support functions. This time the labelling probabilities are represented as the weights of the first layer, and the scene and model measurements in the form of compatibility coefficients constitute the input. The weights are therefore initialised either by the prior probabilities, if the standard MAP rule is used (Section 3.2), or by the probabilities conditional on the attributes if the scheme of Section 8.4 is used. Thus the weights no longer contain the model information; in this respect this representation has perhaps more in common with the Hopfield model [32]. The auxiliary layer and exponential activation function of the representation of Fig. 11.1 are now implicit in the mechanism that uses the support functions of the output to update the weights of the first perceptron layer. The outputs on convergence in this case are not binary; rather they give an indication of the relative supports for the set of labellings. In practice however, after iterating, one labelling support will usually significantly dominate the others.

In Fig. 11.2, the network for a single object is shown. In the standard version of the algorithm, in which the probabilities for all the objects are updated simultaneously, this network would have to be replicated for each object. On the other hand, if the asynchronous updating scheme of 8.1 is preferred, a single copy of the network is sufficient, but the process must be iterated for each object.

The representation of Fig. 11.2 could be extended to include the explicit generation of the compatibility coefficients by means of a layer of radial basis functions.

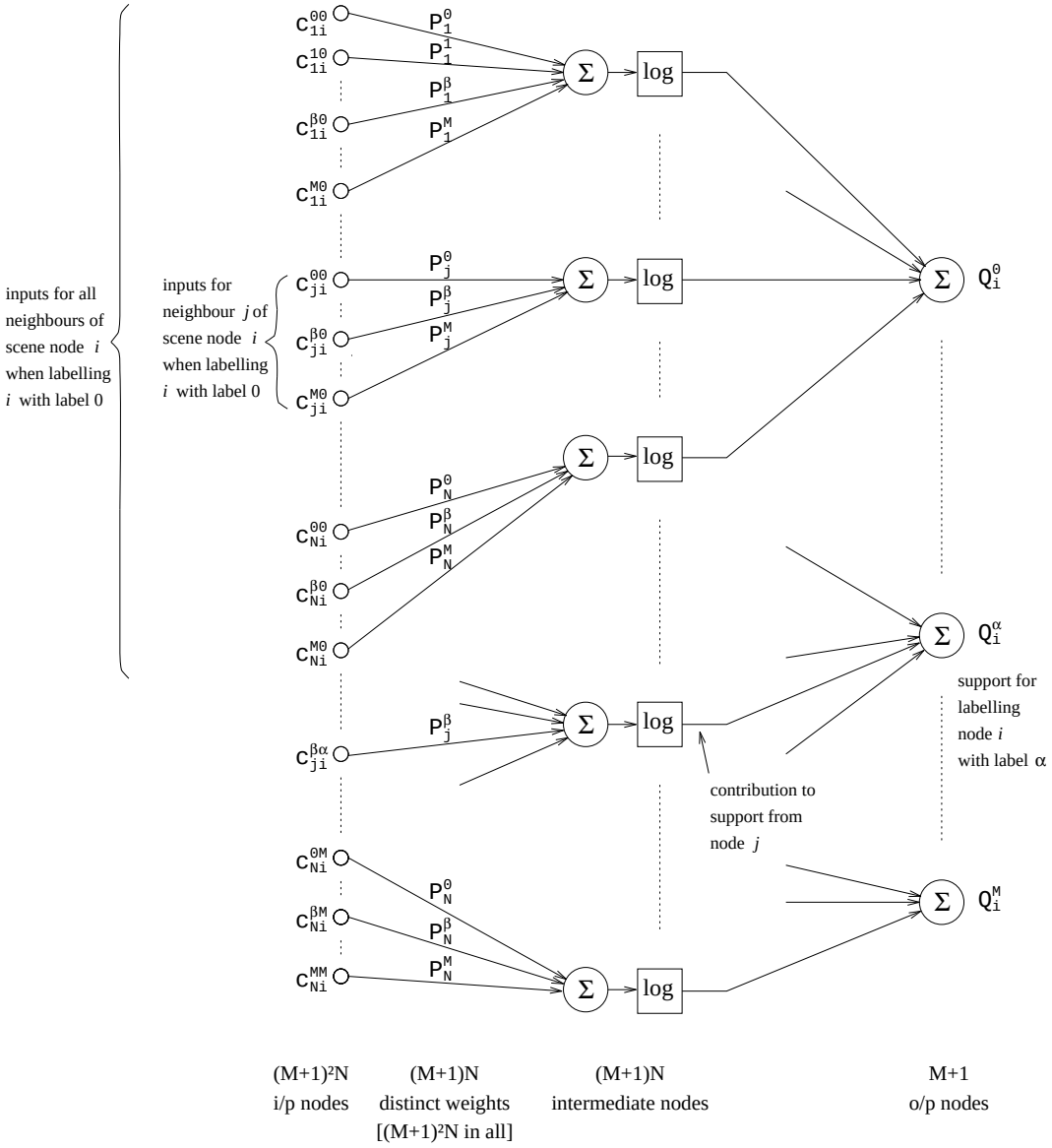


Figure 11.2: An alternative neural net representation of MAP labelling scheme

Chapter 12

Discussion

12.1 A review of the method

In this work, we have developed an algorithm for matching two-dimensional structures composed of geometrical features. We found that the method is reliable, rugged, computationally efficient and readily parallelisable, and for many applications has no arbitrary parameters to adjust.

The algorithm requires the following inputs from the user: the scene and model feature measurements, the total possible range of the scene measurements (for the null p.d.f. values) and the distributions of both the scene measurement errors and the uncertainty in the scene-to-model transformation. Provision of the feature measurements in a suitable form is usually straightforward. The scene measurement range was deduced from the image size and the type of feature being extracted.

The distributions in principle present more of a problem. The measurement error distribution depends in turn on knowledge of both the feature extraction process, which may be awkward to analyse, and the noise model of the original imaging process, which may be difficult to obtain. The transformation error distribution may also be awkward to calculate. Processing the distributions to obtain the required attribute and relation distributions is also usually complicated. In the applications that we have tried so far, we assumed that the attribute and relation distributions were Gaussian; some estimate of the measurement variances had therefore to be provided in order to compute the variances and covariances. When attributes were used, a cutoff value for the attribute distribution was also provided to reduce the number of potential labellings.

The power of the method ultimately stems from two basic decisions. The first was to use the Maximum A Posteriori probability rule (the MAP rule) to determine the correct object labelling. The MAP rule required the direct expression of the measurement information; this in turn led us to the second decision: to express the information in terms of relations based on measurements from pairs of features, which enabled us to disentangle the errors due to measurement noise from the uncertainty in the overall relationship between the scene and model measurements.

The MAP rule

The use of the MAP rule required the evaluation of posterior match probabilities. Because the values of these probabilities were not known directly, Bayes's rule was then used to manipulate them into a form that consisted of quantities that could be computed. In this way, a relaxation rule was created for which all of the terms had a direct physical interpretation, with the consequence that a methodology to evaluate them was apparent. The result is an algorithm that requires no arbitrary parameters to be supplied; for those parameters that are required, a methodology is provided to calculate them either from the process that generated the features from the scene or from properties of the original scene itself.

Use of binary relations

There are two types of unknown quantity in the matching problem: the measurement noise and the uncertainty in the transformation between scene and model. The measurement noise is likely to be substantially independent between the various scene features, whereas the transformation uncertainty is highly correlated. Indeed for many applications the latter can be represented by two or three parameters; these two types of unknowns therefore need to be disentangled. Because the transformation uncertainty can be so concisely represented, it is often possible to find measurement quantities that are invariant to it, and this is the approach taken in this work. The elementary (unary) measurements do not usually have the required invariance; instead measurements from combinations of features (*i.e.* relations) were used that were found to have the required invariance. The number of features involved in a relation should be as low a possible, because the computation time (in the worst case) increases exponentially with this number. For many applications, a set of binary relations (*i.e.* based on pairs of features) could be found that were essentially invariant to the transformation while still adequately representing the structure of the scene and model features. This work was therefore restricted to such binary relations.

We conclude by summarising the principal benefits of the method:

- the derivation of the update rule from the Maximum A Posteriori probabilities indicates how all of the terms in the rule are to be evaluated;
- the algorithm is tolerant of measurement errors, the presence of spurious scene features and the absence of model features;
- a noise model for the measurements is explicitly included; and
- the algorithm converges significantly faster than many relaxation methods, often only needing two iterations, and usually generates a consistent result after a single iteration.

12.2 Philosophers' Corner

Clearly no thesis that is to be submitted for the degree of Doctor of Philosophy is complete without some philosophical discussion. In our case, the discussion topic is (briefly) on the

various interpretations of the term "probability" and hence on the justification of the way in which we assign values to probabilities in this work. There are a number of fervently-held but differing views as to what a probability actually is, although the three basic axioms of probability are accepted by all of them. In [59], Weatherford identifies four broad classes of interpretation that are widely held:

The classical view: This view can only cope with situations in which there is some "atomic" set of events which are finite in number, mutually exclusive and assumed to be equiprobable. The probability of each outcome is then the reciprocal of the number of possible outcomes. The probability calculus then enables us to assign values to various combinations of these outcomes. Particularly importantly from our point of view, it embraces the Principle of Indifference (or the Principle of Insufficient Reason): any two events are to be considered equiprobable if we have no means of deciding which is the more probable. Thus if we are tossing a coin, we assume that the coin is fair, with a probability of 1/2 for each outcome. Even if we know the coin is not fair, unless we know more about its imperfections, we still assume an equiprobable outcome.

The relative frequency view: This is perhaps the most commonly-taught view. In this view some experiments are made; the probability of a particular outcome happening is then defined as the ratio of the number of actual such outcomes to the total number of experiments, as the number of experiments tends to infinity. Of course we can never conduct an infinite number of experiments, so in practice the number has to be sufficiently large to satisfy some criterion. Thus for the coin-tossing experiment, accurate probability values can be found regardless of the fairness of the coin, provided of course that the coin does not wear out in the process.

The a priori view: This view was a development of the classical approach, but attempts to enlarge in a rigorous manner the range of situations that can be handled, beyond those covered by the classical approach. It defines probability as "a measure of the logical support for a proposition on given evidence" [59]. The associated theory attempts to generate useful probability values even when only a small number of experiments have been performed. It does however appear to make harsh judgements about the fairness of the coin if a small number of tosses all produce the same outcome.

The subjective view: In this view, a probability value is what a particular individual believes it to be. Thus it is clearly related to the way people place bets. If I know the person tossing the coin well, I might make my own mind up about the likely outcome, particularly if there is money involved.

The labelling algorithm can be viewed as the combining of a set of probabilities (the prior probabilities), with a set of measurements which are instances of some random variables, to create a further set of probabilities (the posterior probabilities). Thus to implement the algorithm, in addition to providing the measurements, we have to assign values to the prior probabilities, and also establish the distributions of the random variables. Deciding what distribution to use is in effect assigning relative values to the probabilities of all of the possible values of the corresponding random variable. We discuss these two probability assignments in turn.

Prior probabilities

When we assign values to the prior probabilities (Section 5.1), we make (in general) two assumptions. We take some view of the likely proportion of null matches (*i.e.* we make a guess) in order to assign a value to the prior probability of a null match. This seems to correspond to the subjective view of probability. We then use the Principle of Indifference to assign equal values to the remaining probabilities, which clearly corresponds to the classical view.

Probability distributions

Distributions have to be established for the scene attributes and relations, and possibly for the scene-to-model transformation. For attributes and relations, there are two types of distributions (Chapter 5): those that involve a null match and those that do not:

Distributions involving a null labelling: We argued that a uniform distribution was appropriate, largely on the basis once again of the Principle of Indifference. In this case, as we are dealing with a random variable of the continuous type, we are looking at a limiting case of what the classical view can handle, and use arguments that perhaps more properly belong to the subjective approach.

Distributions not involving a null labelling: Here we argued that the precise distribution to be used could be estimated from knowledge of the feature extraction process, pushing the problem further back up the processing path to some point outside the range of the discussion here. What we actually did in practice was to assume a Gaussian distribution, as this was felt somehow to fit in with what was observed from a subjective examination of typical images. Thus in practice we are clearly in the subjective camp again.

Thus we see that the justification for assignment of probability values is largely based on a combination of the classical and subjective views. Clearly we are not performing any exhaustive randomised experiments to determine the values, so the relative frequency view in particular is *not* relevant here. We argue that in all four views have their place, depending on the particular problem; no one of them has an intrinsic "rightness" that some of their proponents would claim. We take the modest successes so far of our labelling method to add weight to this view.

12.3 Future work

There are inevitably many ways in which the work described here can be extended, and the following suggestions by no means form a complete list.

Theoretical analysis

More theoretical underpinning is needed, particularly in assessing the effect of some of the independence assumptions, and in establishing the convergence of the iteration scheme. Some work on the latter aspect has been undertaken by others [55, 56]. If the relation of the scheme to optimisation methods were better understood, perhaps a more rigorous approach could be taken to the decision of when to stop iterating. At present, one or more of the heuristic schemes of Section 4.3 is used.

Attribute and relation types

The types of attributes and relations used at present, particularly the latter, were chosen on a fairly arbitrary basis — they seemed reasonable. A methodology is needed to establish the relative merits of different choices. Alternatively, for computational efficiency, it would be desirable to use a set of relations that had a diagonal covariance matrix.

Higher-order relations

When modelling uncertainties in the scene-to-model transformation, we attempt to select relations that are invariant to the transformation. Because we only consider binary relations at present, the method is limited to transformations to which these binary relations are invariant (or approximately invariant). For example in the case of line segment matching, we are in principle limited to an unknown Euclidean transformation. If higher-order relations (particularly ternary) could be incorporated without undue extra computational cost, the range of applications could be considerably extended.

Extension to "$2\frac{1}{2}$-D" problems

If the use of higher-order relations is infeasible, we may be able to represent the transformation uncertainties as noise. A simple example of this was the incorporation of small scaling errors in Section 6.1. In the applications of Chapter 9 that have a three-dimensional aspect, the method currently ignores the 3-D aspect and treats the problem as one of straightforward 2-D matching. Thus in the stereo matching example (Section 9.3), the effect of the epipolar constraint on the possible feature misalignment was ignored. Similarly, in the example of Section 9.4 where image features were matched to a 3-D model for which the projection parameters are known approximately, no account was taken of the manner in which these parameters propagate through the projection process. In both cases, if the parameters are known approximately, it should be possible to model their uncertainty as part of the scene feature measurement errors.

Improved feature extraction

In a typical machine vision application, the matching algorithm will form just one of many components that make up the system. The algorithm is only as good as the measurement

information that goes into it, and so applications of this method would benefit if more effort were spent on developing improved methods of feature extraction. One might say that the purpose of feature extraction is to reduce the amount of data in the original image to manageable proportions. At the same time, it is important that the minimum possible amount of useful information is lost in the process. If we look at the road matching examples of Fig. 9.1, the representation as line segments is crude, to say the least. In particular the curvature information has been largely lost, and the roads are broken in arbitrary places. In the stereo example of Fig. 9.4, edges are also from time to time broken in the wrong places. These sort of effects represent a degradation of the features, which puts a corresponding burden onto the matcher. The work of Sha'ashua and Ullman [51] deserves further investigation in this respect. A wider range of feature types could also be examined, including in particular corners and junctions.

Feature quality

In the feature extraction process, it would be useful if it were possible to assess the "quality" of the features. This information could be used both to improve the assessment of the prior probabilities (particularly that of the null labelling), and also to improve the measurement noise modelling.

Improved evaluation of the labelling accuracy

Again regarding the algorithm as a system component, it is important that the algorithm should attempt to indicate whether the feature correspondences were likely to be the "correct" ones. At present this is done by examining the overall fit of the matched features using a heuristic least-mean-square approach, and reporting the overall mismatch. The number of null matches obtained and the mean support in practice also give useful information about the quality of the result. A more rigorous approach would be preferable.

Comparison of performance with other methods

As was discussed in Chapter 11, there are many different methods of feature matching. It would be instructive to compare quantatively the performance of the present method with other, more established methods, particularly in respect of robustness, computational complexity and ability to deal with a wide range of applications.

Bibliography

[1] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] H.S. Baird. *Model-based image matching using location*. MIT Press, 1985.

[3] J.R. Beveridge and E.M. Riseman. Hybrid weak-perspective and full-perspective matching. In *Conference on Computer Vision and Pattern Recognition*, pages 432–438, 1992.

[4] B. Bhanu. Representation and shape matching of 3-D objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:340–351, 1984.

[5] B. Bhanu and O.D. Faugeras. Shape matching of two-dimensional objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:137–156, 1984.

[6] E. Bienenstock. Neural-like graph-matching techniques for image processing. In D.Z. Anderson, editor, *Neural Information Processing Systems*, pages 211–235. Addison-Wesley, 1988.

[7] A. Blake. The least disturbance principle and weak constraints. *Pattern Recognition Letters*, 1:393–399, 1983.

[8] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[9] R.C. Bolles. Robust feature matching through maximal cliques. *Proc. Soc. Photo-Opt. Instrum. Engrs*, 182:140–149, April 1979.

[10] K.L. Boyer and A.C. Kak. Structural stereopsis for 3-D vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10:144–166, 1988.

[11] T.A. Cass. Polynomial-time object recognition in the presence of clutter, occlusion, and uncertainty. In *Second European Conference on Computer Vision (Santa Margherita Ligure, Italy, May 1992)*, pages 834–842. Springer-Verlag, 1992.

[12] W.J. Christmas, J. Kittler, and M. Petrou. Matching in computer vision using non-iterative contextual correspondence. Paper in course of preparation.

[13] W.J. Christmas, J. Kittler, and M. Petrou. Matching in computer vision using probabilistic relaxation. To appear in *IEEE Trans. on Pattern Analysis and Machine Intelligence, 1995*.

[14] W.J. Christmas, J. Kittler, and M. Petrou. Analytical approaches to the neural net architecture design. In *Pattern Recognition in Practice IV*, Vlieland, The Netherlands, 1994.

[15] W.J. Christmas, J. Kittler, and M. Petrou. Location of objects in a cluttered scene using probabilistic relaxation. In *2nd International Workshop on Visual Form*, Capri, Italy, 1994.

[16] W.J. Christmas, J. Kittler, and M. Petrou. Matching of road segments using probabilistic relaxation: reducing the computational requirements. In J.G. Verly and S.S. Welch, editors, *Sensing, Imaging and Vision for control and guidance of aerospace vehicles*, volume SPIE 2220, pages 169–179, Orlando, Florida, USA, 1994.

[17] W.J. Christmas, J. Kittler, and M. Petrou. Matching of road segments using probabilistic relaxation: a hierarchical approach. In S-S. Chen, editor, *Neural and Stochastic Methods in Image and Signal Processing III*, volume SPIE 2304, pages 166–174, San Diego, California, USA, 1994.

[18] W.J. Christmas, J. Kittler, and M. Petrou. Non-iterative contextual correspondence matching. In J-O. Eklundh, editor, *Computer Vision – ECCV'94*, volume II, pages 137–142, Stockholm, Sweden, 1994. Springer-Verlag.

[19] W.J. Christmas, J. Kittler, and M. Petrou. Modelling compatibility coefficient distributions for probabilistic feature-labelling schemes. Accepted by BMVC'95, the Sixth British Machine Vision Conference, 1995.

[20] L.S. Davis. Shape matching using relaxation techniques. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1(1):60–72, January 1979.

[21] R. Deriche, R. Vaillant, and O.D. Faugeras. From noisy edge points to 3D reconstruction of a scene - a robust approach and its uncertainty analysis. In P. Johansen and S. Olsen, editors, *Theory & applications of image analysis*, volume 2 of *Series in Machine Perception and Artificial Intelligence*, pages 71–78, Aalborg, Denmark D9108, 1992. World Scientific Publ. Co. PTE Ltd., Singapore.

[22] O.D. Faugeras and M. Berthod. Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3:412–423, April 1981.

[23] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[24] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22:67–92, 1973.

[25] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRF's: Surface reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(5):401–412, May 1991.

[26] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[27] D.E. Gharhaman, A.K.C. Wong, and T. Au. Graph optimal monomorphism algorithms. *IEEE Trans. Systems, Man, and Cybernetics*, 10(4):181–188, April 1980.

[28] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9:469–482, 1987.

[29] E.R. Hancock and J. Kittler. Edge labelling using dictionary-based relaxation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12:165–181, 1990.

[30] E.R. Hancock and J. Kittler. An improved error criterion for neural networks. In P. Johansen and S. Olsen, editors, *Theory and applications of image analysis*, Aalborg, Denmark D9108, 1992. Series in machine perception and artificial intelligence vol.2.

[31] J.J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. In *Proceedings of National Academic Science*, volume 81, pages 3088–3092, USA, 1984.

[32] J.J. Hopfield and D.W. Tank. 'Neural' computation of decisions in optimisation problems. *Biol. Cybern.*, 52:141–152, 1985.

[33] R.A. Hummel and S.W. Zucker. On the foundations of relaxation labeling processes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 5(3):267–286, May 1983.

[34] R. L. Kirby. A product rule relaxation method. *CGIP*, 13:158–189, 1980.

[35] S. Kirkpatrick, C.D. Gellatt, and M.P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

[36] J. Kittler. Relaxation methods and their neural net implementation. In V. Cherkasky and H. Wechsler, editors, *From statistics to neural networks*, Berlin, 1994. Springer-Verlag.

[37] J. Kittler, W.J. Christmas, and M. Petrou. Probabilistic relaxation for matching of symbolic structures. In *Workshop on Structural and Syntactic Pattern Recognition*, pages 471–480, Bern, Switzerland, 1992.

[38] J. Kittler, W.J. Christmas, and M. Petrou. Probabilistic relaxation for matching problems in computer vision. In *Proceedings of the Fourth International Conference on Computer Vision*, pages 666–673, Berlin, Germany, 1993.

[39] J. Kittler and J. Föglein. On compatibility and support functions in probabilistic relaxation. *CVGIP*, pages 257–267, 34 1986.

[40] J. Kittler and E.R. Hancock. Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:29–51, 1989.

[41] C. Koch, J. Marroquin, and A. Yuille. Analog 'neuronal' networks in early vision. In *Proceedings of National Academic Science*, volume 83, pages 4263–4267, USA, 1986.

[42] S.Z. Li. Matching: invariant to translations, rotations and scale changes. *Pattern Recognition*, 25:583–594, 1992.

[43] S.Z. Li, J. Kittler, and M. Petrou. On the automatic registration of aerial photographs and digitised maps. *Optical Engineering*, 32(6):1213–1221, June 1993.

[44] S. Peleg. A new probabilistic relaxation scheme. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2:362–369, 1980.

[45] M. Pelillo. Relaxation labeling processes for the travelling salesman problem. In *Proceedings of 1993 International Joint Conference on Neural Networks*, volume 3, pages 2429–2432, 1993.

[46] M. Pelillo and M. Refice. An optimization algorithm for determining the compatibility coefficients of relaxation labeling processes. In *ICPR*, volume B, pages 145–148, 1992.

[47] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labelling processes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16:933–945, 1994.

[48] M. Petrou, P.L. Palmer, W.J. Christmas, and J. Kittler. Robust skeletonization and object recognition from grey images. In *2nd International Workshop on Visual Form*, Capri, Italy, 1994.

[49] B. Radig. Image sequence analysis using relational structures. *Pattern Recognition*, 17:161–167, 1984.

[50] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Trans. Systems, Man, and Cybernetics*, 6:420–433, June 1976.

[51] A. Sha'ashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *International Conference on Computer Vision*, pages 321–327, 1988.

[52] L.G. Shapiro and R.M. Haralick. Structural description and inexact matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3:504–519, September 1981.

[53] F. Fogelman Soulie, B.Lamy, and E.Viennet. Multi-modular neural network architectures for pattern recognition: Applications in optical character recognition and human face recognition. *IJPR*, 1993.

[54] F. Stein and G. Medioni. Structural indexing: efficient 3-D object recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14:125–145, 1992.

[55] A.J. Stoddart, M. Petrou, and J. Kittler. On the foundations of probabilistic relaxation with product support. In course of preparation.

[56] A.J. Stoddart, M. Petrou, and J. Kittler. Probabilistic relaxation as an optimizer. Accepted by BMVC'95, the Sixth British Machine Vision Conference, 1995.

[57] S. Ullman. Relaxation and constraint optimization by local process. *Comp. Graph. and Image Proc.*, 10:115–195, 1979.

[58] N.M. Vaidya and K.L. Boyer. Stereopsis and image registration from extended range features in the absence of camera pose information. *Conference on Computer Vision and Pattern Recognition*, 76, 1982.

[59] R. Weatherford. *Philosophical foundations of probability theory*. Routledge & Kegan Paul, 1982.

[60] W.M. Wells. MAP model matching. In *Conference on Computer Vision and Pattern Recognition*, pages 486–492, 1991.

[61] A. Witkin, D. Terzopoulos, and M. Kass. Signal matching through scale space. *International Journal of Computer Vision*, 1:133–144, 1987.

[62] K. Yamamoto. A method of deriving compatibility coefficients for relaxation operators. *Comp. Graph. and Image Proc.*, 10:256–271, 1979.

[63] B. Yang, W.E. Snyder, and G.L. Bilbro. Matching oversegmented 3-D images to models using association graphs. *Image and Vision Computing*, 7:135–143, 1989.

[64] S. Zucker and J. Mohammed. Analysis of probabilistic labeling process. *IEEE PRIP Conf. Chicago, USA*, pages 167–173, 1978.