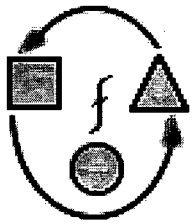

FACE DETECTION BY FACETS:
COMBINED BOTTOM-UP AND TOP-DOWN SEARCH USING COMPOUND TEMPLATES

by

GLENDON R. HOLST

B.Sc. Honours, The University of British Columbia, 1997

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(Department of Computer Science)

We accept this thesis as conforming to the required standard

David G. Lowe

James J. Little

THE UNIVERSITY OF BRITISH COLUMBIA

June 2000

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of COMPUTER SCIENCE

The University of British Columbia
Vancouver, Canada

Date July 10th, 2000

DE-6 (2/88)

Abstract

As detection domains increase in size and complexity, new techniques are needed to effectively search the image and feature space. In this thesis, I explore one such approach to object recognition in the domain of face detection. This approach, dubbed compound templates, is compared to a single template approach. The developed system, Facets, provides an implementation of both techniques to enable fair comparison.

The compound template technique uses subfeatures and spatial models to represent a compound object (such as a face). From these compound models, hypothesis-based search then combines top-down and bottom-up search processes to localize the search within the image and feature space. Detected subfeatures become evidence for facial hypotheses, which then guide local searches for the remaining subfeatures based upon the expected facial configuration.

The compound technique is described and a comparison of the compound templates technique with a single template technique in a mug-shot style face domain is presented. A description of the implementation, along with issues surrounding the compound templates approach is also provided. Attention is paid to performance, including both efficiency and accuracy. The results are complex; but the strengths, weaknesses, and various trade-offs of the two techniques are detailed.

The combined bottom-up and top-down approach of compound templates demonstrates a clear advantage over bottom-up only approaches. The compound templates approach also demonstrates better performance for feature sparse images, detection accuracy, domain coverage, and for domains with increasing size.

Table of Contents

List of Tables

List of Figures

Acknowledgements

To Lisa Jackson for such a good start.

To my grade 8 science teacher, for the best answer possible.

To the amazing residents of St. John's College, for enriching discoveries, restored faith in humanity, and hope for the future.

To Hayedeh Behzad... you know! ;-)

To David Lowe for patience, supervision, and proofreading.

1 Introduction to Face Detection

Efficient and effective object recognition techniques are needed for increasingly large and complex image domains. One such candidate approach was implemented and tested for the domain of face detection. This thesis presents the result of that work.

1.1 The Domain

Faces are diverse, semi-rigid, semi-flexible, culturally significant, and part of our individual identity. As a domain they are already well studied and many image databases already exist. For complexity the domain is extensible along a continuum ranging from expressionless frontal, to increasingly varied expression, to increasing pose variance, to increasing artistic interpretation.

Face recognition systems [Konen], [Brunelli], [Beymer] already exist for a variety of purposes, from surveillance to mug shot queries. Many of these systems use eigenfaces (principal components); however, eigenface systems need to first normalize the face image. Normalization requires a priori location of the face and its subfeatures. Face detection systems could locate the face and subfeatures in the image as a preliminary step for face recognition. Both face recognition and face detection are categorization problems, except that recognition categorizes faces by individual, and detection categorizes by face or non-face.

The chosen domain is face detection of photographic, grey-scale, near frontal, mildly expressioned, faces. This domain is practical yet interesting; challenging yet tractable. Examples of the domain follow[1].

---

[1] The CMU 1104 image comes from the online testing page for the CMU Face Detection system. The URL is: http://www.ius.cs.cmu.edu/IUS/usrp0/har/FaceDemo/images/1104/input.gif

The Yale images are available from: http://cvc.yale.edu/projects/yalefaces/yalefaces.html

The Nottingham images are available from: http://pics.psych.stir.ac.uk/cgi-bin/PICS/New/pics.cgi

Figure 1 - The CMU 1104 image contains 400 images of 40 different people, in 10 poses each. These faces formed the training set.

Figure 2 - The Yale image database contains 165 faces from 15 different people, in 11 poses each. These faces were part of the testing set.

Figure 3 - The Nottingham image database contains 100 faces from 50 women and 50 men. These faces were part of the testing set.

1.2 Related Work

There are many other approaches to the face detection problem. Some techniques rely on whole face templates or models for detection [Sung], [Lanitis], [Rowley], others rely on facial subfeatures [Viola], [Yuille], [Takacs]. A variety of detection techniques are employed, from correlation [Brunelli], neural nets [Rowley], creseptrons [Weng], eigentemplates [Shakunaga], [Sung], [Viola], Bayesian models [Viola], and flexible models [Lanitis], [Yuille]. Some approaches use bottom-up search [Weng], others use top-down search [Sung], and still others combine both search types [Shakunaga]. Combining bottom-up and top-down processes appears as a promising way to guide the search efficiently. There are

other systems in other domains which appear to use this approach successfully [Matsuyama], [Milanese].

1.2.1 Single Template

Kah-Kay Sung and Tomaso Poggio propose a technique for face detection [Sung] which uses a small set of full face templates, and a small set of non-face templates. This approach uses 'view-based' models, created by the system from training examples, hence the templates are essentially model images. As described in their paper, "[They chose] a piece-wise continuous modelling scheme because face patterns appear to occupy a smoothly varying and continuous region in the vector space — i.e., more often than not, a face pattern with minor spatial and/or grey-level perturbations still looks like another valid face pattern." [Sung] At 19x19 pixels the templates are small, striking a balance between discrimination power and computational efficiency. Though sufficient in size to distinguish a face, they do not contain sufficient detail to determine subfeature expression (e.g., direction of gaze is not modeled in the template views). The 12 view-based models are the prototypes and anti-prototypes that describe the face space. Each prototype is encoded using the 75 largest eigenvectors.

The process of detection is a classification of each window of the image, at different resolutions, as one or none of the prototypes. The window starts off at 19x19 pixels and grows to 100x100 pixels by a factor of 1.2 each time. The window is then scaled down to 19x19, equal in size to the prototypes, and converted into an eigenface. A distance measurement is then taken between each of the 12 prototypes and the image window. The classifier is a Multi-Layer Perceptron (MLP) which learns to map these distance measurements to a yes/no face classification. Essentially, these distance measures either place an image near a face prototype, near a non-face prototype, or not near a face prototype. If only the first case is true a face is detected, otherwise not.

The distances themselves are a 2-value concoction of the normalized Mahalanobis distance in the 75 top eigenvector space between the window image and the prototype, and the Euclidean distance between the window image and its projection onto the eigenface. The Mahalanobis distance takes into account the higher variance of faces along the direction of the eigenvectors so that a measured distance between the centroid of a

prototype and an image becomes shorter for images which, while keeping the same Euclidean distance from the prototype centroid, lie closer to the higher variance eigenvectors (i.e., the boundary of equal distance to a prototype is not circular, as in Euclidean distance, but oval). The prototypes themselves are automatically generated by a k-means clustering algorithm from a database of examples. One database contains the normalized canonical faces, the other contains non-face examples, discovered in the process of testing the system.

Sung and Poggio claim that their technique is a general purpose feature detector, stressing that "[their] ultimate goal is to propose a general methodology for taking on feature detection tasks in multiple domains, including industrial inspection, medical image analysis and terrain classification, where target patterns may not be rigid or geometrically parameterizable, and where imaging conditions may not be within the user's control" [Sung]. How would their system behave as the domains are expanded, (e.g., with multiple poses, or articulated objects), and what happens to system performance as the number of domain classes increases? How could we expand the system to detect subfeatures which require higher resolution templates than the enclosing feature, or combine this technique with other feature detection techniques, such as deformable templates [Yuille], better suited to a subdomain of the larger domain? It is these sorts of questions that motivated and informed the exploration into compound templates. Although not as sophisticated as the above eigentemplate approach, simple templates were used as the comparative norm in the Facets implementation. The combined and compound approach described in chapter 1, section 3.1, has the flexibility to combine image analysis techniques, and to recognize articulated objects.

1.2.2 Compound Representations

On the question of multiple pose, David Beymer provides one solution as part of his face recognition under varying pose system. The feature finder in this system looks for both irises and a nose lobe. The search is performed on a 5 level image pyramid, with level 0 being the original image, and level 4 the lowest resolution version. Search begins at the top level, using 30 full face model templates that cover 5 rotations, 3 image plane rotations, and 2 scales. At each level, matches are determined by correlation scores greater than a predetermined threshold. At levels 3

and 4, all matches are found and then sorted by correlation score. Search then proceeds in a depth first manner from the level 3 hypothesis. The first match at level 0 wins. At each level more templates are introduced, covering the pose space using smaller intervals and providing higher resolution templates. Not all the templates are used, rather just the small neighbourhood of templates surrounding the rough face hypothesis from the previous level. On the top levels the templates are whole faces, but they become templates for the individual features, eyes and nose, on the lower levels.

One benefit is that the combinatorial explosion of potential faces is prevented while still benefiting from a larger number of feature exemplars. Instead of using eigenfaces as in Sung's approach, templates and image windows are compared using normalized correlation on gradient, Laplacian, and original grey-level versions. Recognition performance for the original grey-level images was 94.5%, and above 98% accuracy for the preprocessed images. [Beymer]

Attractive features of Beymer's approach are the use of face hypotheses to guide further search, and the use of subfeatures within the model framework. The search process proceeds from general to specific, in a decision tree like fashion. As in the Beymer approach, the posited technique uses the location of the more general features to guide the search for lower level prototype matches. Likewise, the incorporation of subfeatures continues the search for more specific prototypes, but in a sub region. How would the system efficiency degrade as new domain objects were added? If the higher level prototypes perform their discrimination tasks well (i.e., there are fewer paths from the tree root to a specific low level prototype) then the search space might only grow logarithmically, otherwise the performance could be worse. It seems plausible, however, that the domain model space would grow much faster than any space of subfeatures, or image primitives. Subfeatures provide a compact way to describe the superfeature space, limiting the combinatorial explosion that affects the superfeature space.

Subfeatures also allow for localized search of simpler features. A top-down approach, such as those discussed here, may benefit from the slower growing subfeature space; however, if the superfeature space is very large, having many non-generalizable top-level configurations, then many initial searches are required. Each of these searches may benefit from

the hierarchical nature of the search (from superfeature to subfeature, or from general to specific) but without a priori knowledge the search must cover the entire superfeature space. One way to provide the a priori knowledge is to use bottom-up search. First searching for the subfeatures provides information about the superfeatures in the image space; however, if the search is performed for all subfeatures, there is no need for the top-down search, and thus no way to benefit from it. Such a complete bottom-up search includes more of the subfeature space than the top-down search would require. Combining the bottom-up and top-down searches while preserving their benefits would seem a worthwhile goal.

1.2.3 Combined Search

Matsuyama and Hwang's SIGMA system detects houses, roads and driveways from aerial images. It must deal with a top-level model space (superfeature space comprising the connection of roads with driveways with houses) where the possible spatial and orientational configurations are so enormous, no reasonable set of image level templates could holistically describe every possible scene. In dealing with the complexity of this domain, SIGMA incorporates bottom-up and top-down processes in a complementary way. SIGMA first looks for subfeatures of the domain, combine these together into a coherent upper-level view and then searches for missing features. Extending Beymer's approach so that it combines both bottom-up and top-down processes, first searching for facial sub-features (eyes, nose, mouth), requires that issues such as false positives, undiscovered features, overlapping hypotheses, and the enforcement or discovery of spatial relationships are dealt with. In short, resolving issues that the SIGMA approach addresses.

The SIGMA system comprises three experts, the Geometric Reasoning Expert (GRE), the Low-Level Vision Expert (LLVE), and the Model Selection Expert (MSE), which correspond respectively to the three knowledge categories of: scene domain knowledge, image domain knowledge, and meta-knowledge mapping the two knowledge domains. There are also databases for domain model classes and the runtime instances and hypotheses. Detection first proceeds with a segmentation process searching prominent features. This task is initiated by the MSE which selects objects from the domain database and provides the LLVE with the object's description needed to perform the actual image analysis. The

LLVE reports back the image level description of the found item (or possibly a Not-Found message). The MSE converts the image level description into a scene level object instance and places it in the runtime database. The GRE evaluates the object instances in the runtime database and determines, using the domain models, which other objects are related to the runtime instance. The GRE then instructs the MSE to search for the missing instances. When and if these new objects are found, the MSE again instantiates them and places them in the runtime database where the GRE establishes the relationships between them (e.g., PART-OF or spatial-relationship). The GRE is also responsible to ensure that the runtime database only contains consistent and coherent hypotheses called interpretation networks. Several interpretation networks may exist in the runtime database at the same time, but they form independent and competing scene interpretations of the image. The quality of the interpretations is determined by its size — assuming here that larger networks require more image level structure to support them and hence are less likely random artifacts. In the course of detection, interpretation networks may be joined, split, duplicated, or removed according to rules in the GRE which work to remove interpretation conflicts and support coherent interpretations. The iterative interaction of these three expert modules detects features and creates a scene level interpretation. [Matsuyama]

This process can deal with false positives, undiscovered features, spatial relationships, plus provide additional benefits. False positives are unlikely to become part of the final interpretation since it is unlikely that they could form coherent relationships with enough features to compete with the correct interpretation. The use of top-down analysis aids in the detection of, as yet, undiscovered features. The top-down expectations encourage the use of local and complete image analysis techniques of a type that would be computationally expensive to blindly perform on the entire image. The LLVE uses threshold and binarize at different threshold levels to discover a feature, and such trial and error attempts are localized thanks to the top-down process. Deformable template techniques [Yuille] [Lanitis], which provide feature parameterization at a level of detail difficult to achieve with template classification, work best if placed near the feature (e.g., deformable eye templates work best if placed slightly below the actual eye) and top-down placement would support these requirements while removing the computational expense of unfocused search. The bottom-up process also reduces computational

cost since the GRE does not search for domain objects that do not have image level evidence. Another benefit comes declaring spatial relationships in a more flexible and independent manner than a strictly image based approach. Since the SIGMA operative model separates scene and image domains, it is possible to have 3D scene domains and reason about their appearance in the 2D image domain and vice versa — a flexibility that view-based models do not afford. The scene level provides a more natural interface for the designer who can express the domain using the concepts they are familiar with.

Although the hypothesis generation, evaluation, and reasoning aspects of SIGMA add complexity to the system, they do provide a flexibility and elegance from the combined bottom-up, top-down processes they support. Whether this combination of search processes produces a more efficient or more accurate system, in comparision to other approaches, is unknown. Thus, part of the motivation for this thesis is to compare a combined and standard approach.

1.2.4 Compound and Combined Face Detection

The aerial image domain of SIGMA is more complex than the face domain proposed here. Is the corresponding complexity of the SIGMA system warranted in the face domain? The compound technique proposed here is envisioned as generalizable to other domains and other image analysis techniques; the face domain and the Facets implementation is but a single instance in the exploration. Anyway, Takeshi Shakunaga, Keisuke Ogawa, and Shohei Oki use combined bottom-up and top-down search in their face detection system [Shakunaga].

Shakunaga's system uses 32x64 pixel templates to represent eyes, eyebrows, ears, nose and mouth center. Each face is represented by these eight subfeatures. The subfeatures are represented in a combined eigenvector space and subfeature detection proceeds in a way similar to the Sung design[2]. The first phase of detection searches for all subfeatures, collecting those above a given threshold. The second phase enumerates 10 feasible combinations of found instances. A 3D face model is created for each combination, and its pose parameters are estimated

---

[2] The Shakunaga system doesn't have non-subfeature prototypes, and instead of a MLP (Multi-Level Perceptron) for classification it uses a threshold based on the Mahalanobis distance.

based on subfeatures and defaults. The third phase performs a local top-down search for any missing subfeatures. Missing subfeature attributes are calculated from the 3D face model and its pose. If a new subfeature instance is detected, the face model parameters are re-calculated, and phase three repeats until no more subfeatures are found, or all subfeatures are found. Although the first search phase searches for all features it discards the poor performers to limit the combinatorial explosion of face hypotheses. The top-down phase then searches locally for these discarded subfeatures, improving the correct detection rate from around 65% to near 92%.

1.2.5 Motivations

Some single template techniques [Sung] are advertised as generalizable to larger domains. Of the compound techniques [Beymer], or those combining bottom-up and top-down [Matsuyama], [Shakungaga], the emphasis appears on detection performance. I was interested to see what efficiency merits a combined and compound approach would have compared to a single template approach. I was also interested to extend my understanding of the implementation issues involved in a combined approach. To this end I developed the Facets face detection system as a platform to compare both approaches in equivalent implementations.

1.3 Overview

The compound template technique is an instantiation of the more general compound and combined approach outline below. Some key issues of the general approach are discussed, followed by a description of the compound template face model and methodology.

1.3.1 General Approach

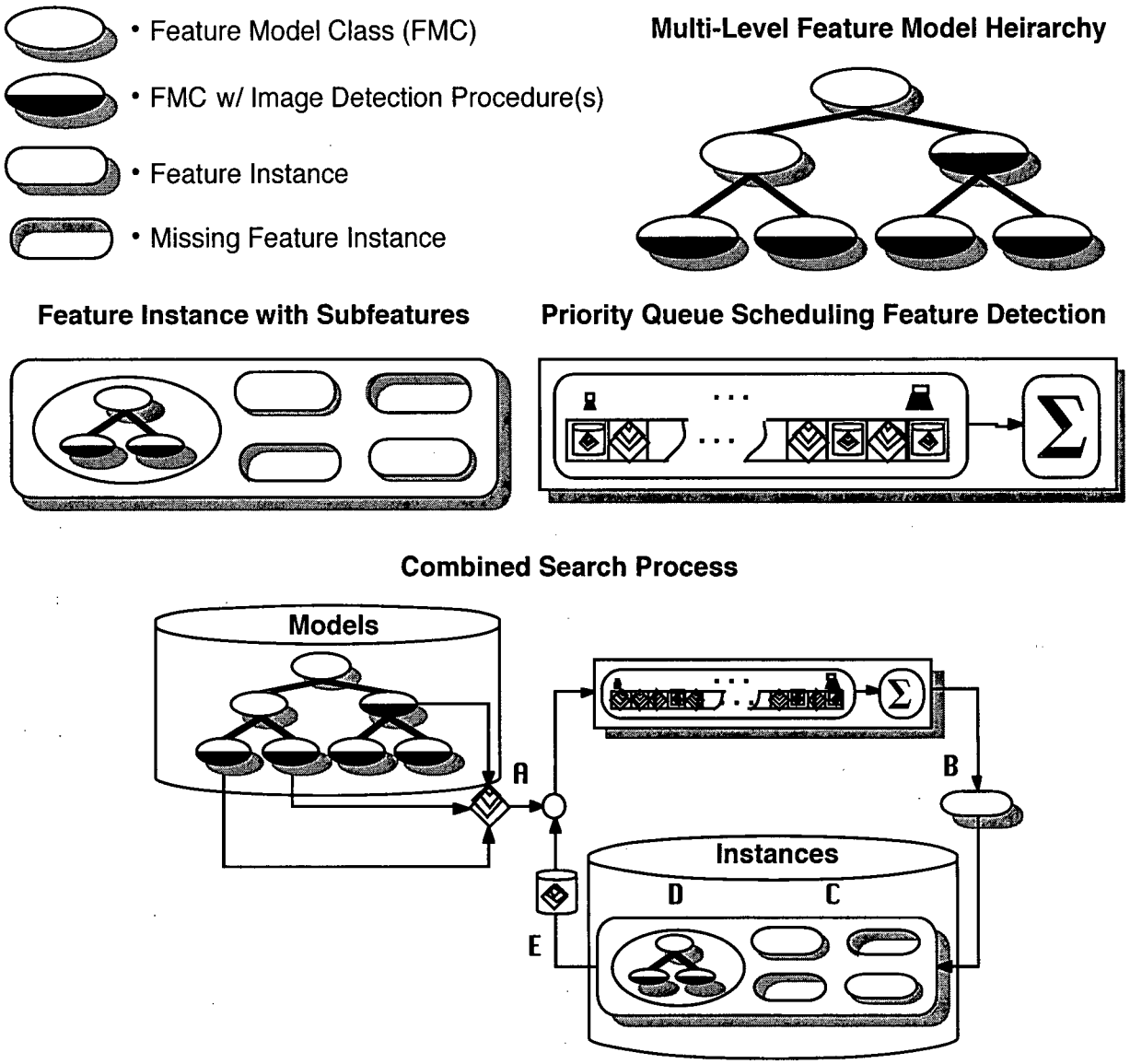

Figure 4 - Compound and combined technique requirements: Illustration (top) of multi-level structure and legend for Combined Search Process (bottom). The feature instance is shown with four subfeatures (two found, two missing), and the model used to calculate feature model parameters. Illustration of Combined Search Process are 7

In general terms, a compound and combined technique requires:

1 An object model with a multi-level structure and partitioning subfeatures. Two or more levels are required to invoke a combined bottom-up and top-down search. There may be any number of object models.

2 At a minimum, the base subfeatures, (those features which do not themselves have subfeatures), require an associated image detection

procedure. Other features may also have associated image detection procedures, but this is optional. The image detection procedure may implement any image analysis technique.

3 An invocation queue for search requests. This could be a FIFO queue (essentially a direct call to the image detection procedures) or it could be a priority queue. As shown later, the queue helps weave the bottom-up and top-down processes together, and provides the designer with considerable flexibility.

The combined search process for the compound technique performs as follows:

A The search begins by invoking the image detection procedures for some initial subfeatures (usually base subfeatures). The choice of initial subfeatures critically affects accuracy and efficiency. This is discussed in more detail below.

B When a feature is found, a model instance for each related superfeature is created, and the detected subfeature is given to the superfeature model instance as evidence. Optionally, if the detected subfeature fits within the hypothesis space of an existing superfeature model instance, it is given to the existing instance. Managing overlapping hypotheses is discussed in more detail below.

C A model instance represents a hypothesis about an object (or object-space) supported by evidence from the image. The model instance contains the subfeature instances already found (which provide evidence for the hypothesis), and the estimated feature space for all subfeatures (required for missing subfeatures).

D The model instance may also estimate model level parameters and instance states. A superfeature model instance may set the estimated feature space, or the model instance may calculate them itself.

- Parameters, which might represent rotation, scale, and hypothesis strength, are calculated based upon the evidence (subfeature instances), potential defaults, and model requirements.
- Instance states might include the status of subfeature searches, or whether the model instance represented an object instance or object-space. A model instance may only represent the hypothesis for a single object instance, or it may represent an object space. In the later case, the model may choose the best object instance when the search completes.

E When a model instance reaches a critical threshold of evidence or

strength (normally the first subfeature), the top-down phase begins. The image detection procedures are called for the missing subfeatures using the estimated feature space as the search parameter. Search scheduling issues are discussed below. Avoiding duplicated search is covered as part of the discussion on overlapping hypotheses, below.

F A model instance completes its search either when it has exhausted the subfeature search space, or it represents a complete object instance. The entire search completes when the initial bottom-up search completes, and all model instances have completed their search.

1.3.2 General Issues

The efficiency and accuracy of a combined search depends upon the subfeature space used for the initial bottom-up search. If the initial search comprises the entire subfeature space then the search becomes essentially equivalent to a pure bottom-up search. If the initial feature space is not sufficient to find at least one subfeature per object (or whatever the threshold may be), then not all the objects in the image will be detected. Using a scheduling queue provides a way to increase the feature space of the initial search without delaying the top-down phases. The discrimination quality of the initial subfeatures, and the computational resources needed to detect those features are also important considerations. For subfeature s, detected instance s_i of subfeature s, and object o_{si} containing subfeature s in configuration i, the preference is for subfeatures with a higher probability $P(o_{si} | s_i)$. The preference is also for subfeatures with efficient image analysis procedures. The design decisions made in the Facets system are provided below.

Initial bottom-up search does not require base subfeatures. A mid-level subfeature might have better discrimination powers and lower computational cost. To illustrate, imagine a face model that has eyes, nose, and mouth as mid-level subfeatures. The eye subfeatures are models which contain subfeatures for iris, pupil, lashes, and whites. The mouth subfeature is a model with subfeatures for teeth, lips, and tongue. Searching for a face by first searching for irises, pupils, lashes, eye whites, teeth, lips, and tongues makes little sense, since such subfeatures might not exist in the image, or are not easily discriminated. Searching for these subfeatures is best done after the enclosing feature

is localized. For this reason it is not necessary to initiate the bottom-up search from the base subfeatures when intermediate subfeatures provide for more efficient and accurate detection.

When subfeature instances are detected they are evidence for some object level hypothesis space. It is very likely that overlapping hypotheses are created, especially if the initial subfeature search space is complete enough to ensure the detection of all objects. It is important to merge similar model instances before they instigate their top-down search. There are many ways to perform the overlap detection and merging, and the Facets system provides rudimentary facilities for this, described later.

The scheduling queue provides a way to prioritize the image level search. For example, in the initial bottom-up search, the queue could prioritize based upon the discrimination capabilities and the resources required for the subfeatures. As mentioned, this would provide a way to weave in the top-down searches, performing them before all the bottom-up searches completed. The queue could also let the designer separate the search into temporally distinct stages (e.g., bottom-up stage, first top-down stage, second top-down stage, etc.)

Aggregate searches allow additional search control logic that depends on several search results, for example, cancelling or delaying the search for one subfeature if another subfeature was not detected.

1.3.3 Compound Templates

The compound template technique is a type of compound and combined method. For a model it uses a 2D plane with subfeature points and four degrees of freedom: rotation in the image plane, scale, and <X,Y> location. The image analysis procedure for subfeatures uses normalized correlation. In this implementation there are four subfeatures partitioning the face and the correlation is performed on the grey-scale image using grey-scale image templates.

A template specifies the spatial arrangement of sub-features. A 2D model also specifies spatial arrangement, but with more flexibility. With whole templates, the entire template is searched, even if initial portions of the template are poor matches. It would appear beneficial to perform a

faster initial search with smaller template sub-regions and then complete the search in the areas denoted most promising. Using a model with template sub regions (subfeatures) gains the flexibility of the model, occlusion support, combinatorial savings for the object representations, plus control over the search process.

The compound template representation for a compound face uses four subfeature types: left eye, right eye, nose, and mouth.

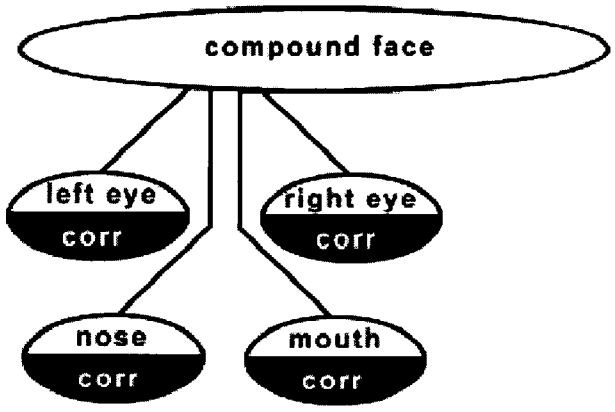

Figure 5 - Compound face model with four subfeatures. Subfeatures use normalized correlation for the image analysis procedure.

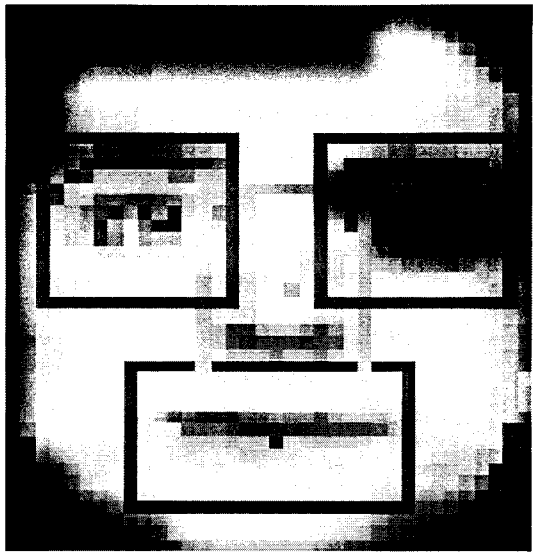

Figure 6 - Typical facial subfeatures

Each subfeature type contains a set of masked templates used by the normalized correlation image detection procedure. The mask describes a, possibly, non-rectangular region of interest in the template that is used for comparisons. A subset of these templates is specified as initial search candidates. A complete search, over the entire image, is performed for these templates. When the correlation score for a template is greater than the threshold, the corresponding subfeature instances are created. As features are found and instantiated, compound face instances are created to represent the face hypothesis evidenced by the subfeature instances. Based on the 2D model and the subfeature evidence, the face instance requests localized searches for the remaining features. The strength of the initial evidence determines which localized searches are performed first.

The evidence strength for a subfeature instance is the correlation score for the template match. The strength for a compound face is partly the evidence strength of its contained subfeatures, and partly the inverse of

the distance from the current configuration (spatial, orientation, and size) of the subfeatures compared to the ideal model.

Implementation details, such as managing overlapping hypotheses, compound face strength calculations, the initial subfeature space, aggregate search optimizations, and more, are discussed below in the Techniques chapter.

1.3.4 Concurrent Interests

The Facets system, besides serving its primary goal as a testbed to compare two face detection approaches, served as an exploratory tool for my other interests. Sometimes these interests informed Facets beneficially.

I find systems like SIGMA [Matsuyama], and Copycat [Hofstadter] interesting because they create their own models during the perceptual process. SIGMA uses declarative rules to specify valid connections of roads, driveways, and houses. The system then uses the image to inform the development of the scene model, and the scene model to inform the analysis of the image. Copycat creates a model of an analogical mapping by combining existing relationship concepts. The generation process is intriguing in that it results from the interaction of many parallel Codelet agents which both add and remove relationships from the mapping model. The parallel Codelets attracted my interest in agent oriented processes, and the dynamic model generation meshed with my interest in prototype languages.

However, the attempt at declarative model generation was replaced by the simple 2D model described later. While the object model of Facets supports runtime generation of classes, much of its use was removed during code profiling. Such runtime support was not needed by the models used. Thread overhead from attempted parallelism was detrimental to performance, especially for the smaller localized searches. This was removed, and now only one search occurs at a time (although aggregate search optimization provides some of its benefits). Searches are still prioritized, though.

Interest in anytime algorithms informs the implementation of compound templates. The state of a face model instance is always updated with the

most recent information available. Stopping the detection process yields all faces found so far with the best combination of subfeatures yet found.

2 Techniques for Facets

This chapter details the core techniques used along with the Facets system implementation.

2.1 Simple Templates

The foundation of the Facets face detection system is the simple template. A simple template is composed from a pyramid of masked grey scale images. Detection is performed using normalized correlation on the source image pyramid.

Facets reads any image format supported by QuickTime® and its own Visage format. Visage is an object repository for images, templates, pyramids, filters, manifests (list of objects and their search attributes), and their attributes. Images are stored and represented in grey scale as an array of real numbers in the range 0..1, where 0 is black and 1 is white. Images allow associated named attributes. These are used to store image angle and scale attributes. Templates are represented as two images of the same size. One image is the source, the other is the mask. The mask has a one-to-one pixel correspondence with the source and represents the degree to which a source pixel is included for correlation comparisons. In the mask, 0 represents the excluded source and 1 represents full inclusion.

Facets provides image scaling and rotation filters. Rotation is performed using bilinear interpolation. Scaling is calculated using the weighted average of all pixels in the source image that overlap the destination pixel. The weight is based on the percentage area overlap between the destination pixel and the source pixel (their intersection).

For search efficiency, both templates and images are represented at various scales in image pyramids [Burt]. Detection target images are scaled from the bottom to top (largest to smallest). The original image is the largest (bottom - image 0) image, and the smallest (top - image N) image is close to 64 pixels in size. Each image X in the pyramid, where X >=1, is 0.75 the area of the image X-1 below it. Scaling is performed from the original image each time (i.e., image 1 is 0.75 the size of image 0, image 2 is 0.5625 the area of image 0, etc.). For example, a training set image (containing 25 faces) would create a pyramid with a 235x285 pixel

bottom image, a 7x9 pixel top image, and containing 25 levels, numbered 0..24.

Similarly, pyramids are used for feature templates. To create template pyramids, the source image and the mask are scaled by the same amount, and the scaled templates form the template pyramid. The scaling ensures that the maximum change in width or height for the scaled template is 2 pixels smaller than the preceding template in the pyramid. This ensures that templates are discretized to within a pixel of their neighbours, as required for correlation. Each scaled template is created from the original largest template. The area of the bottom and top template is specified when generating the pyramid. For faces the range is 400 pixel area for the bottom template and 64 pixels for the top, producing 7 levels average. For eyes the range is 150 pixels for bottom and 12 pixels for top, with 5 levels average. For noses the range is 200 pixels for bottom and 25 pixels for top, with 6 levels average. For mouths the range is 200 pixels for bottom and 12 pixels for top, with 8 levels average. The 400 pixel area for the largest face template was chosen as a 20x20 face similar to Sung's 19x19 face templates. The other values were determined by initial experiments to find good working values. Template pyramids may also include rotated templates[3].

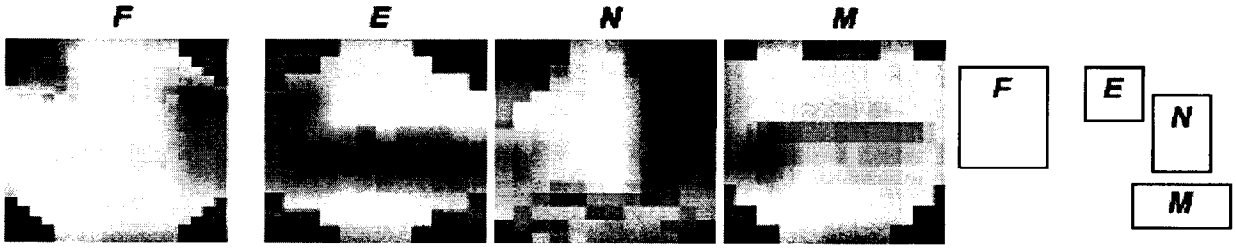

Figure 7 - Feature templates with masks (left) and relative size (right). **F** is face, **E** is eye, **N** is nose, and **M** is mouth.

[3] Although the Facets system supports rotated templates and pyramids, rotation was not used for testing since the domain contained only close-to-upright faces.

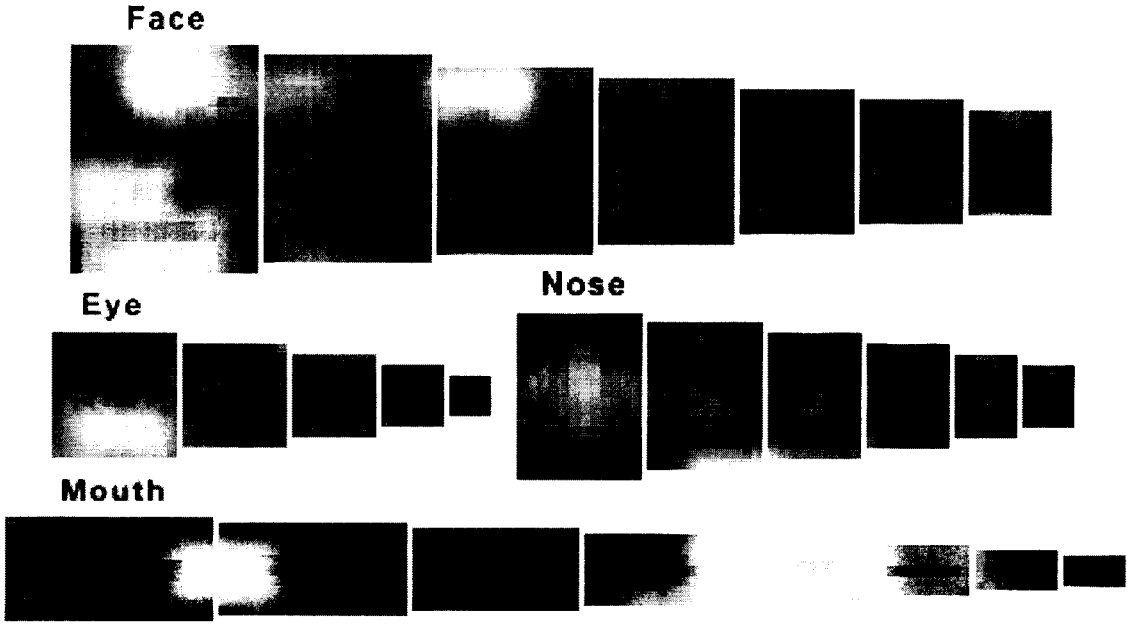

Figure 8 - Template pyramids to scale for face, eyes, nose, and mouth (masks not shown).

Normalized correlation [Moore] is used to detect a template in an image. For a source image S_I, a feature template F_T, and a location L of F_T overlapping S_I, the intersection of S_I and F_T are I, T and M. I is the area of the image S_I contained within the intersection. T is the image area of the template F_T contained within the intersection. M is the mask area of the template F_T contained within the intersection. I, T and M have the same size with length n. I_i is the ith pixel of I. T_i is the ith pixel of T. M_i is the ith pixel of M. $\overline{I}$ and $\overline{T}$ are the means of I and T respectively. They are calculated as:

$$\overline{I} = \frac{1}{n}\sum_{i}^{n} I_i \;,\quad \overline{T} = \frac{1}{n}\sum_{i}^{n} T_i$$

σ_I and σ_T are the standard deviation of the I and T vectors respectively. They are calculated as:

$$\sigma_I^2 = \frac{1}{n-1}\sum_{i}^{n}\left(I_i - \overline{I}\right)^2 \;,\quad \sigma_T^2 = \frac{1}{n-1}\sum_{i}^{n}\left(T_i - \overline{T}\right)^2$$

The masked correlation r lies in the range [-1,1] and is calculated as:

$$r = \frac{\sum_{i}^{n} \left(\frac{I_i - \overline{I}}{\sigma_I} \right) \left(\frac{T_i - \overline{T}}{\sigma_T} \right) M_i}{\sum_{i}^{n} \left[\left(\frac{I_i - \overline{I}}{\sigma_I} \right)^2 + \left(\frac{T_i - \overline{T}}{\sigma_T} \right)^2 \right] M_i}$$

The correlation r approaches 1 as the positive linear association between I and T increases. The correlation threshold of r >= 0.7 is used to determine valid template matches. Expanding the equations for variance σ^2_I and σ^2_T, and correlation r produces formulas calculatable in a single pass:

$$\sigma^2_I = \frac{1}{n-1} \left[\sum_i^n I_i^2 - \frac{1}{n} \left(\sum_i^n I_i \right)^2 \right], \quad \sigma^2_T = \frac{1}{n-1} \left[\sum_i^n T_i^2 - \frac{1}{n} \left(\sum_i^n T_i \right)^2 \right]$$

$$S_I = \sum_i^n I_i, \quad S_T = \sum_i^n T_i, \quad S_M = \sum_i^n M_i,$$

$$S_{MI} = \sum_i^n I_i M_i, \quad S_{MT} = \sum_i^n T_i M_i, \quad S_{MIT} = \sum_i^n I_i T_i M_i,$$

$$S_{MI^2} = \sum_i^n I_i^2 M_i, \quad S_{MT^2} = \sum_i^n T_i^2 M_i$$

$$r = \frac{2\sigma_I \sigma_T \left[S_{MIT} - \frac{1}{n} S_I S_{MT} - \frac{1}{n} S_T S_{MI} + \frac{1}{n^2} S_T S_I S_M \right]}{\left[\sigma^2_T \left(S_{MI^2} - \frac{2}{n} S_I S_{MI} + \frac{1}{n^2} S_I^2 S_M \right) + \sigma^2_I \left(S_{MT^2} - \frac{2}{n} S_T S_{MT} + \frac{1}{n^2} S_T^2 S_M \right) \right]}$$

Repeated summation terms, S_*, are optimized into a single calculation and masked templates precalculate constant subexpression values (i.e., those which don't include I). These values are used when the template intersection is the entire template; which occurs in the majority of cases. They are calculated during correlation otherwise. Another optimization occurs for masks with a large number of zeros Z. If the cost of testing ($M_i == 0$) n times in the correlation loop is less than the cost for Z inner loop calculations involving M_i, then inner loop calculations involving M_i are not performed when $M_i == 0$.

The feature space for a template[4] represents the range of sizes, rotations and locations that would contain a feature in the image. Feature spaces are composed from: the minimum and maximum area range in base coordinates, the rotation angle range, and the enclosing rectangle for the template center in base coordinates. Base coordinates are in pixels based upon the original (i.e., largest) target image. Facets provides for a rudimentary representation of feature space regions. Feature space regions are just collections of feature spaces. If two feature spaces overlap sufficiently they are represented as a single enclosing feature space. Feature spaces sufficiently overlap when the area and angle ranges overlap, and the intersection I_{12} of the overlapping enclosing rectangles R_1 and R_2 is greater in area than $[area(U_{12})-area(R_1)-area(R_2)+area(I_{12})]$, where U_{12} is the smallest rectangle enclosing R_1 and R_2.

Template search proceeds in three stages: candidate search, refine search, and instance search. Searches are performed on a per template basis and require a feature space argument FS[5]. The candidate search stage searches the given feature space FS using the lowest resolution templates and images possible from the respective pyramids. The search space is

---

[4] Templates and template pyramids are used interchangeably to represent a feature instance.

[5] There is a limit on the minimum area for feature space FS that is based upon the area range covered by the template pyramid. The minimum area ensures that detected features are not too small, and that templates used to create an instance have sufficient resolution. For subfeature templates, the minimum area is equal to a template in the middle of the pyramid. For face templates, the minimum area is equal to a template a quarter the way from the bottom of the pyramid. Compared to the size of the original feature, maximum subfeature template sizes are bigger than their face template counterparts, so the difference in the minimum area factor equalizes the two.

covered by traversing the target image pyramid from top (small) to bottom (large). At each level L the feature space FS_L for that level is calculated in base coordinates. The template pyramid is queried for a set of templates T that cover the FS_L space. The feature space covered by T should equal FS_L within tolerances, otherwise a gap-error results (i.e., one of the pyramids has too few levels). Templates T are matched to the target image I_L at level L using correlation. FS becomes $FS-FS_L$ and the process repeats for the next level L-1 or stops when FS is empty or after level 0. If FS is not empty after searching each image pyramid level, then a gap-error results. The result for the completed search is a feature space region (possibly disjoint) of discovered matches suitable for the next search stage.

The refinement stage searches the given feature space on ever lower target image pyramid levels (i.e., increasing image size). This differs from the candidate search in that FS is not updated to remove the searched space FS_L. Search stops when level 0 of the pyramid is reached, or when templates T cannot cover the space FS_L. This process localizes the feature location more accurately and uses higher resolution templates for better discrimination. The result is a feature space region of the most localized (higher resolution) match suitable for the next search stage. The final instance search stage searches the given feature space starting from the bottom of the target image pyramid[6]. The first detected match becomes the evidence for a feature instance. The feature class owning the template (face, left eye, right eye, nose, mouth) creates a feature instance. The strength of a feature instance is the correlation r of the detected match, plus an early detection factor[7].

Feature templates are stored in a Visage file called the environment. Each environment has a manifest which details each templates' information, including feature class, initial detection, skip refine step, instance candidate, default weight, and evidence sets. A template belongs to a feature class: face, left eye, right eye, nose, or mouth. Templates

---

[6] Some work from the refine search stage is duplicated during the instance search stage.

[7] The early detection factor is calculated as `0.001/detection_time`, where `detection_time` is an integer, greater than 0, measuring system ticks in 1/60th of a second. The purpose of this factor is to 'bless' equivalent matches detected earlier. Equivalent matches can occur when search spaces overlap.

with initial detection set to true are invoked at the start of face detection (i.e., they are feature templates which do not require previous evidence). The refine search stage for a feature template is skipped if skip refine step is true. For best performance, this was set to false for all templates. Templates with instance candidate set to true can generate instances, otherwise the last search stage is skipped and matched template features can only be evidence to invoke other searches. For the tests, all feature templates could create instances. The default weight is a real number from 0..1 that represents the a priori strength of a given feature template. The default value for testing was 0.8, which means that only candidate matches stronger than 0.8 will pre-empt the search for initial feature templates. The evidence set contains a list (possibly empty) of other feature templates to search for when detection matches are found for this feature template[8]. The feature space to search is given by the found detection match.

To start face detection, the user must first open a Visage file containing the desired database of feature templates and set it as the environment. The manifest is read, the feature templates loaded, and the feature models created with the associated feature templates. It is only necessary to perform this step once, although for testing the environment was reloaded each time[9]. The user then selects a target image pyramid to perform the detection on. A candidate search over the entire image is invoked for all feature templates with initial detection set. All instances created are collected for later use and analysis.

For face feature instances, the search process is complete now. This describes the simple template technique used as the basis for comparison. For left eye, right eye, nose, and mouth subfeature instances, the search process is just beginning. When a left eye, right eye, nose, or mouth class creates a subfeature instance it gives the instance to the compound face class which creates a compound face instance to contain the given subfeature. The compound face instance is responsible to generate a hypothesis space from the subfeature evidence and continue the search.

---

[8] This feature was not used during testing.

[9] To limit the effect of memory fragmentation.

2.2 Compound Face Model

Compound face instances are objects containing data and functionality.

Data

Compound faces need to keep track of their detected subfeatures (**Subfeature Collections**, explained below), and those subfeatures which actually comprise the face (**Primary Subfeatures**). To limit duplicate searching, compound faces also need to record the feature spaces already searched (**Searched Spaces**). The behaviour of the compound face instance depends upon its current state (**Detection State**), and the system performance is measured by the time required to completely detect a face (**Last Modified Time**, explained below).

- **Subfeature Collections:** The subfeature collections SC_{left_eye}, SC_{right_eye}, SC_{nose}, and SC_{mouth}, where each collection contains the subfeature instances of the designated type added to the compound face instance. `NIL` represents the nonexistent subfeature instance. Each combination of $<sf_{le}, sf_{re}, sf_{n}, sf_{m}>$ where,
  - $sf_{le} \in SC_{left\_eye} \cup \{NIL\}$,
  - $sf_{re} \in SC_{right\_eye} \cup \{NIL\}$,
  - $sf_{n} \in SC_{nose} \cup \{NIL\}$,
  - $sf_{m} \in SC_{mouth} \cup \{NIL\}$,

  represents a hypothesis instance, and the set of all combinations $\left\{ <sf_{le}, sf_{re}, sf_{n}, sf_{m}> \middle| sf_{le}, sf_{re}, sf_{n}, \& sf_{m} \text{ each} \in SC_{*} \cup \{NIL\} \right\}$ is the hypothesis space HS_{cfi} for the compound face instance `cfi`. From the HS_{cfi} the compound face only hypothesizes the existence of a single face (i.e., a single compound face instance cannot represent the potential that two or more faces exist in the image).

- **Primary Subfeatures:** The primary subfeatures PF_{left_eye}, PF_{right_eye}, PF_{nose}, and PF_{mouth}, where
  - $PF_{left\_eye} \in SC_{left\_eye} \cup \{NIL\}$,
  - $PF_{right\_eye} \in SC_{right\_eye} \cup \{NIL\}$,

- $PF_{nose} \in SC_{nose} \cup \{NIL\}$,
- $PF_{mouth} \in SC_{mouth} \cup \{NIL\}$,

and NIL represents no subfeature instance. A primary feature represents at most a single subfeature instance (i.e., only 1 or 0 in number). The primary hypothesis instance $<PF_{left\_eye}, PF_{right\_eye}, PF_{nose}, PF_{mouth}>$ represents the most likely face hypothesis instance yet found (e.g., a compound face with four primary instances would represent a single complete face). Primary instances are determined by maximizing the hypothesis strength generated by the underlying 2D face model.

- **Searched Spaces:** The searched spaces SS_{left_eye}, SS_{right_eye}, SS_{nose}, and SS_{mouth}, which are the sets of feature spaces searched for each subfeature type. These are used to determine the equivalence of HS_{cf1} and HS_{cf2} for compound faces cf1 and cf2, which assists duplicate hypothesis management.

- **Detection States:** The detection state DS_{cfi} of the compound face instance cfi. DS_{cfi} denotes the states: {dormant, active, finalized}. dormant specifies inactive instances, usually during the construction phase. active specifies the search phase of the instances. finalized specifies that the search was completed.

- **Last Modified Time:** The last modified time LmT. Whenever the primary hypothesis instance changes for a compound face, the LmT is updated with the current system time. LmT is used to determine the detection time for a compound face instance.

Functions

A compound face instance needs to determine its pose from the collected subfeatures (pose estimate), determine the feature spaces for missing subfeatures (feature space estimates), and calculate how well the subfeatures match the estimated pose (hypothesis strength). A method to add subfeatures to a compound face instance is required, and the face needs to manage this collection (accept subfeatures). The compound face must also invoke searches for missing subfeatures (top-down search).

- **Pose Estimate:** Estimate the facial pose. For a given hypothesis instance $<sf_{le}, sf_{re}, sf_n, sf_m>$, the face model, described below, calculates:
  - The rotational angle θ_z of the face,
  - the scaling factor S,
  - and the face center point C in base coordinates.

  The face pose estimate FP for given hypothesis instance HI is denoted FP_{HI}. The calculation is an ad hoc estimate based upon the weighted combination of:
  - The rotation attributes of the sf_{le}, sf_{re}, sf_n, and sf_m subfeature instances.
  - The sizes of the subfeatures comprising the hypothesis instance.
  - The spatial arrangement of the subfeature instance pairs.

- **Feature Space Estimates:** Estimate the feature space containing subfeatures. Feature space estimate $FSE_{[T,FP,HI]}$ is generated for subfeature type T based upon pose FP_{HI} and the face model. The complete feature space estimates $FSE_{[FP,HI]}$ are defined as the set $\{FSE_{[left\_eye,FP,HI]}, FSE_{[right\_eye,FP,HI]}, FSE_{[nose,FP,HI]}, FSE_{[mouth,FP,HI]}\}$ The estimates for the primary hypothesis instance are used to initiate the top-down search for missing (i.e., NIL) primary subfeature instances.

- **Hypothesis Strength:** Calculates the hypothesis strength and estimated potential. The hypothesis strength for a given hypothesis instance $HI = <sf_{le}, sf_{re}, sf_n, sf_m>$ is calculated as a combination of the:
  - Feature strengths of sf_{le}, sf_{re}, sf_n, and sf_m (zero for missing subfeatures).
  - The number of subfeatures found (i.e., not NIL).
  - The distance in feature space from the face model under FP_{HI} to the subfeatures sf_{le}, sf_{re}, sf_n, and sf_m.
  - And a bonus for similar sized eyes.

  Estimated potential is essentially hypothesis strength, except that missing subfeatures for which no search was performed are counted as perfect matches (i.e., the assumption is that the ideal subfeature instance will be found). Estimated potential determines search

priorities, while hypothesis strength determines the best candidate face when detection completes.

- **Accept Subfeatures:** Subfeature instances are added to the appropriate subfeature collection when:
  - The compound face is first created. The subfeature instance which provided the evidence for an enclosing face, is added.
  - During hypothesis management. If a newly created compound face instance `cf1` represents an equivalent hypothesis to an existing compound face instance `cf2`, then all the subfeatures in `cf1`'s subfeature collections are added to `cf2`.
  - A subfeature instance is created as the result of a top-down search. The created subfeature instance is added directly to the requesting compound face instance.

  When a subfeature instance is added, two steps are performed. The first step removes duplicates from the subfeature collection, and the second step updates the hypothesis instance. Removing duplicates (i.e., those subfeatures representing an equivalent feature space) limits the potential number of new hypothesis instances (i.e., limits combinatorial explosions), which is an important consideration for the update step. Duplicates with lower strength are removed. If the newly added subfeature instance `Si` is still an element of the subfeature collection, then all hypothesis instances which contain `Si` are evaluated and compared to the primary hypothesis instance. The hypothesis instance $HI = \langle sf_{le}, sf_{re}, sf_n, sf_m \rangle$ with the greatest hypothesis strength becomes the primary hypothesis instance, and the primary instances are likewise updated $PF_{left\_eye}=sf_{le}$, $PF_{right\_eye}=sf_{re}$, $PF_{nose}=sf_n$, and $PF_{mouth}=sf_m$.

- **Top-Down Search:** Initiate a request to search for missing subfeatures. To complete a search request for compound face instance `cfi`, the detections state DS_{cfi} must be `active`. The search request proceeds as follows:
  - Determine the estimated feature spaces for subfeatures by calling the function described above. The top-down search is partitioned into two phases. In the second phase the features spaces are enlarged by a factor of 1.25 times.
  - For each missing subfeature of type `T` (i.e., PI_T = `NIL`), request a search package from the `T` feature class. The provided package

lists the templates available for searching[10] and includes the requested feature space.

- The requested feature space for the package is added to the searched space SS_T.
- The packages are combined into an aggregate search package. The priority of this search package is set to the expected potential of the requesting face instance.[11]
- The aggregate search package is added to the scheduling queue.

The first search is requested when the DS_{cfi} becomes `active`. The second search is requested after the first search completes, if any subfeatures are still missing.

Compound face instance `cf1` is equivalent to compound face instance `cf2`, if for each subfeature type `T`, all subfeature instances SI_T of type `T` in `cfi`'s subfeature collection SC_T are within the searched space SS_T for `cf2` or within the feature space estimate $FSE_{[T,FP,HI]}$ for `cf2` with primary hypothesis instance `HI`. The equivalent hypothesis comparison is performed during duplicate hypothesis management that is performed after a compound face instance is created, but before it is activated. If `cf1` is equivalent to `cf2`, then all subfeatures collected by `cf1` are added to `cf2`, and `cf1` is removed.

---

[10] The templates marked for initial detection are excluded because the system is already searching for them over their entire feature space and the entire image.

[11] Facets supports other search modes, including: individual feature templates with priority set to the template's default weight, and single docket with priority set to the expected potential.

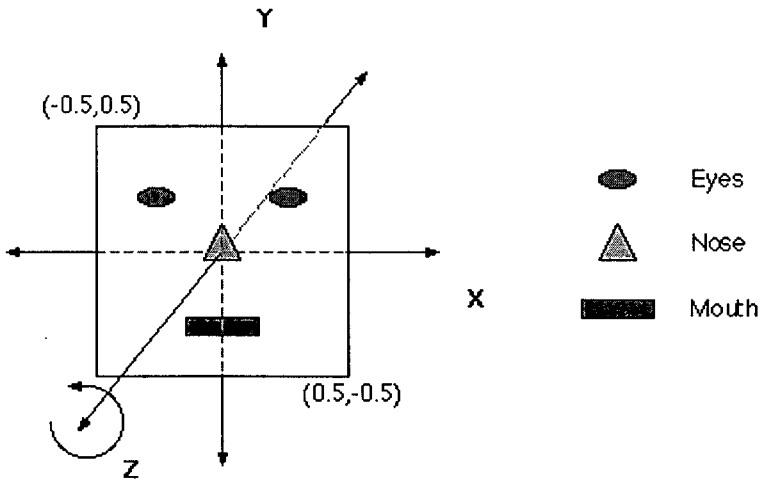

Figure 9 - 2D Face Model. Described below.

The face model is represented on a 2D plane with 4 degrees of freedom: The rotational angle θ_z, the scaling factor `S`, and the face center point `C`.

The plane in model coordinates is a `1x1` square with the point `(0,0)` at the center, `(-0.5,0.5)` at the top left corner, and `(0.5,-0.5)` at the bottom right corner. The left eye position is $(-0.275,0.2\overline{27})$, the right eye position is $(0.275,0.2\overline{27})$, the nose position is $(0,0.\overline{09})$, and the mouth position is $(0,-0.\overline{27})$. The feature centers may vary by `±0.125` in the `X` and `Y` directions. Eyes have a minimum area of `0.06` and a maximum area of `0.28` in model coordinates. The nose has a minimum area of `0.06` and a maximum area of `0.35` in model coordinates. The mouth has a minimum area of `0.07` and a maximum area of `0.4` in model coordinates. The width/height ratio for faces is fixed to `10/11`.

The way in which the compound face instances use the face model to generate face hypotheses and calculate the hypothesis strength was described above as part of the `Pose Estimates` and `Hypotheses Strength` functionality.

2.3 Localized Search

Facets provides three search modes: single templates, template collections, and aggregate search. The initial bottom-up search uses the simple template search mode. For testing, the localized top-down search uses the aggregate search. For each search mode there is a feature detection class. The feature detector for single template preforms the simple template detection described in chapter 2, section 1 above. The

feature detector for collections invokes the single template detector for each template in the collection. The feature detector for the aggregate mode performs optimized searches.

Aggregate feature detectors provide several optimization options; two of which are discussed here. The optimizations work by decomposing search into stages (passes), rescheduling poor performers, and limiting the total search required. Optimization option op_1 is the mid-level technique, while op_2 is the most advanced and complex optimization option provided. Option op_1 performs complete simple template detection as described in chapter 2, section 1, while option op_2 splits template detection into the initial search (candidate search), and the final search (refining and instance search) and reasons about the candidates from the initial search to determine how to proceed with the final search. When template detection is split into initial and final search, the feature detector sorts the candidate matches by strength (correlation score) to order the final search.

The first pass of optimized search determines if it is possible to find a complete face. Each collection, which contains the templates for a subfeature type, is searched for a single candidate subfeature. For op_1 this entails a subfeature instance, for op_2 this entails at least one candidate match with correlation $r >= 0.8$, (possibly more than one candidate found). If none are found, then the search is rescheduled for a later phase. For op_2, final search is then performed for all found candidates. All subfeature instances created during detection are added to the invoking compound face instance. If subfeature collections SC_{left_eye}, SC_{right_eye}, SC_{nose}, and SC_{mouth}, are all non-empty, and the hypothesis strength is greater than 0.65, then the feature detector tries to immediately complete the compound face; otherwise, the search is rescheduled for a later pass. Rescheduled searches are re-prioritized at a fraction of the hypothesis strength, depending on why the search was rescheduled. For example, if one of the subfeature collections SC_T is empty then the search is rescheduled at 25% of the hypothesis strength, otherwise if it just failed to meet the 0.65 strength threshold, it is rescheduled at 75% of the hypothesis strength.

The completion process only searches until $size(SC_T)=N_T$, where N_T is the maximum number of subfeature instances needed for subfeature type T.

For op_1 the values are $N_{left\_eye}=7$, $N_{right\_eye}=7$, $N_{nose}=5$, $N_{mouth}=7$. op_2 uses the values $N_{left\_eye}=3$, $N_{right\_eye}=3$, $N_{nose}=2$, $N_{mouth}=4$ for the subfeature instances, but uses the op_1 values for the number of candidate matches to search for. The candidate matches are sorted in order of higher correlation scores before searching for subfeature instances.

The remaining passes are invoked for rescheduled searches. For op_1 the final pass attempts to complete the face. For op_2, the second pass appears similar to the first except that the initial search is performed for candidates with $r >= 0.75$. If this search fails, the search is rescheduled, otherwise it attempts the completion process (no threshold). Pass three for op_2 is similar to op_1's pass two.

The search passes work well because most compound face instances that do represent actual faces in the image have a high hypothesis strength after the first subfeatures are found. It is important to collect several subfeature instances of each type, since subfeature instances that make for the best face hypothesis instance, may not be the strongest subfeatures individually. The compound face removes duplicate subfeatures from its collection, so the small number of subfeatures found are distinct; even without searching all subfeature templates. op_2 was much more sensitive to the threshold parameters than op_1; however, since op_2 performed better than op_1, it was used for testing. The benefits of aggregate search require a level of coordination that the individual searches cannot provide.

When all passes complete the invoking compound face instance is notified. The face instance may attempt a second search over a larger feature space for any missing features. The priority of the second search is handicapped by 50%.

2.4 System Runtime

Previously, the key system components and their implementations were described. This subsection describes their interaction as an entire system. The runtime is composed from four main units: Template and Model Database (TMD), Runtime Instances (RI), Process Scheduler (PS), and Feature Detection (FD). See Figure 10 for an illustration.

The TMD unit contains the object model classes for detection. The models in this unit are created when a Visage is chosen as the environment. The simple feature classes contain their respective templates from the environment Visage. The compound face class, depicted in Figure 6, has the eye, nose, and mouth simple feature classes as subfeatures, and contains the 2D face model. Compound face instances call upon the compound face class to perform calculations and search requests on their behalf.

The RI unit contains all the instances created during the detection process. These instances are available to display detection results (see Data and Analysis section). The Feature Instances (FI) vector contains all instances. The Compound Face Instances (CFI) vector contains only compound face instances, and it used to search for compound face instances with duplicate hypothesis spaces. If a compound face instance `cfi` does not duplicate an existing instance in the CFI, then cfi is added to both the CFI and the FI.

The PS unit queues detection requests by priority weight, and passes them along to the FD unit when a detector task becomes available. The Facets architecture supports multiple threaded detectors, but for testing only a single detector is used.

The FD unit contains the detectors that perform the feature searches. The target image is set when detection commences. The detection tasks communicate with the feature classes in the TMD unit (e.g., create a new feature instance from a candidate match).

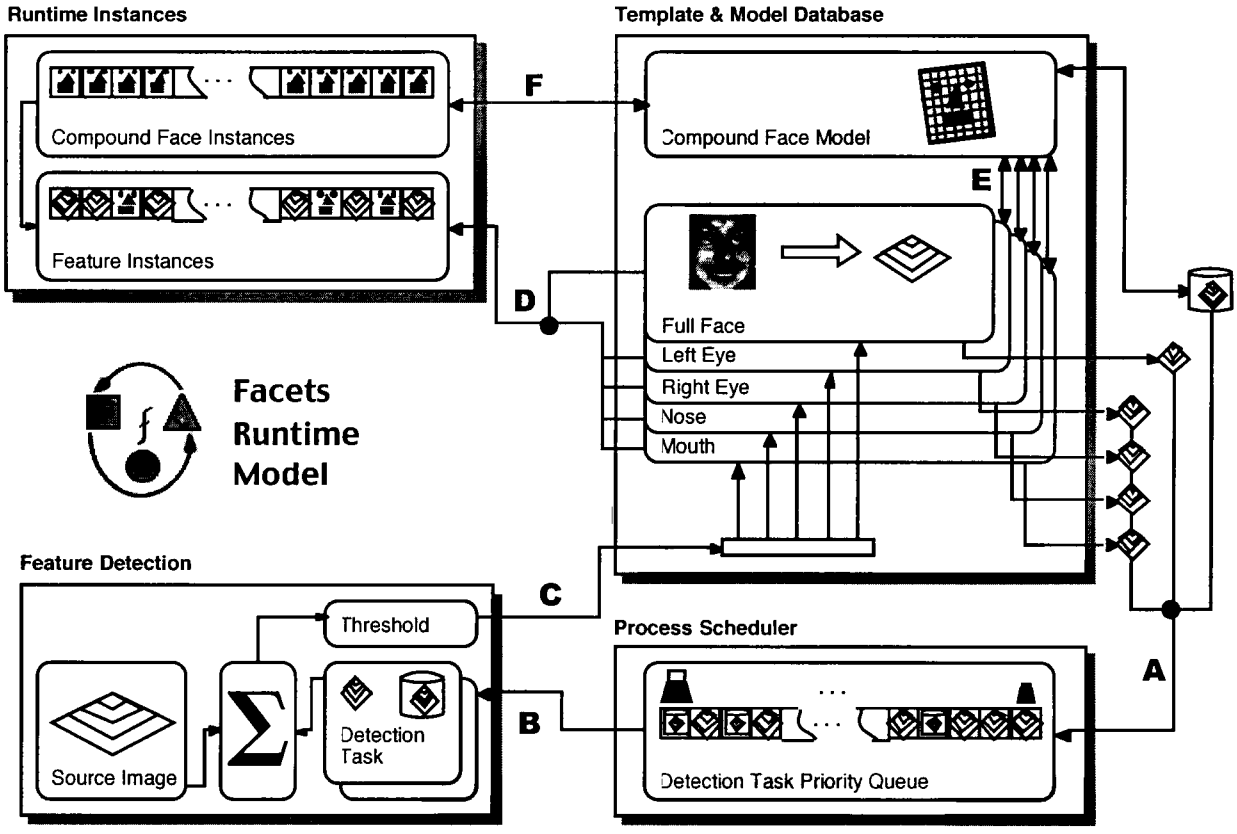

Figure 10 - Facets Runtime Architecture.

During the runtime, as shown in Figure 10, requests for searches are sent (A) by the feature classes to the scheduler, where they are prioritized by value. Detection tasks are sent (B) to the detection unit to search for matches in the specified regions. Matches exceeding the threshold are returned (C) to the original template class. A subfeature instance is created and added (D) to the runtime instances. If the original search request came from a compound face, the instance is returned (E) to the face, otherwise a new compound face instance is created. Compound face instances are placed (F) among the runtime instances; but, to prevent searching overhead from multiple similar hypotheses, similar faces are merged. Compound face instances make localized search requests (F) via the compound face class, which communicates (E) on to the subfeatures for a search, and then makes the request (A) using an aggregate search package.

When the search concludes (or when the user stops the search), the instances in the RI unit are compared against other overlapping features of the same type using hypothesis strength for comparison. Those

instances with the best score and marked as best instances. The best instances are displayed as the detected features.

2.5 Future Implementations

Facets as a system has evolved considerably from optimistic conception to pragmatic conclusion. Many implementation avenues were explored; some successful, some dead ends. Certainly, I am quite sure that were I to create Facets now, I could create a more elegant, robust, and generalizable system in much less time. These are some of the lessons learned.

Good infrastructure is important. Facets benefited from the image processing and representation primitives in its Vista2 library[12], a flexible object database, and object garbage collection (reference counting in Facets)[13]. Although implementing this infrastructure provided good experience and solutions tailored to the system needs, I would have saved considerable effort by investing some time initially into finding existing and more general solutions.

It is worth investing time in automation. Although the Visage object file format supports arbitrary attributes, I did not include any face information with the target images. Originally I thought that I would only perform one testing run, and that the effort needed to gather 50 face templates, and determine accuracy was less than that needed to specify 400 faces and the infrastructure to use this information. As it turned out, I would restart the testing runs several times as issues were discovered and corrected. In the end, some AppleScript scripting facilities were added to the program to automate data collection over the training set (I still had to manually review the detection result images for accuracy though). Currently, model and system parameters were manually set to working values. Automation would support better optimization for these parameters.

---

[12] Vista2 is an object oriented implementation of a subset of the Vista image processing library by Art Pope and David Lowe <http://www.cs.ubc.ca/labs/lci/vista/vista.html>.

[13] The garbage collection supports object sharing without complicating object lifetime and ownership issues. It also reduces dependence, allowing the designer more freedom to experiment and make system changes. As a result, stability was excellent throughout Facet's entire development life cycle.

Create the model first. The model is central to representing and reasoning about the scene-space of an image; and, the capabilities of the model can then guide requirements for the underlying detection system and the feature space. In combination with the automation tactics described above, it then becomes possible to test and even train the model over the training set, even before the detection system is available.

Create a general and robust feature space region representation, independent of the detection techniques. Facets already has a rudimentary feature space representation; however, in some cases its simplicity proved a limiting factor. Some potentially useful features of feature space regions would be: probability distributions over the feature spaces, and set operations (e.g., union, intersection, and subtract). Facets supports combining feature space regions, however, the resulting region can become significantly larger that the union of the two regions. Lack of subtraction (removing a region) in the Facets representation also results in duplicated search effort.

3 Data and Analysis

This chapter details the tests performed using the Facets system implementation described previously.

3.1 Template Database Acquisition

Template databases were created for both simple templates (single face), and compound templates (four subfeatures); and, several tests and comparisons performed. System performance over template acquisition was itself one of the tests performed. The remaining tests used the complete databases. Detection accuracy was determined by visual inspection. Other data was calculated by the system.

The template databases for the simple and compound technique were collected from the CMU 1104 image set shown in Figure 1. The CMU 1104 training set includes 16 images with 25 faces each from 5 people, comprising 400 faces from 40 people in 10 poses all together.

For the simple technique, the whole face templates were collected in the following order:

- An initial face was chosen from the 1st image, a template created from it, and a detection run performed.
- The person with the lowest number of correctly detected faces, in the image with the lowest number of correctly detected faces was selected. From the remaining undetected faces for the selected person, the most normal appearing face was chosen, and a template made from it.
- Another detection run was performed, and the collection process repeated until all faces in the training set were correctly detected.

For the compound technique, the subfeature templates were collected in the following order.

- An initial face was chosen from the 1st image, templates created for all four subfeatures, and a detection run performed.
- For each partially detected face (i.e., the face had one, two, or three subfeatures detected), a template was created from one of the missing subfeatures. Another detection run was performed, and this

step repeated until all potentially correct faces detected had four subfeatures[14].

- After the last detection run, the person with the lowest number of correctly detected faces, in the image with the lowest number of correctly detected faces was selected. From the remaining undetected faces for the selected person, the most normal appearing face was chosen, and a subfeature template created from one of the subfeatures[15].
- Another detection run was performed, and the collection process repeated until all faces in the training set were potentially correct with four subfeatures.
- In a final step, subfeature templates were collected from faces with less than four correctly detected subfeatures. This step was repeated until all faces in the training set were correctly and fully detected.

To collect a template for the database (i.e., create a template pyramid), the face is selected from the image creating a single template. To create a whole face template pyramid, the mask is applied to the template, the template pyramid is then created, and finally saved into the database Visage. To create the subfeature template pyramids, the subfeatures are selected from a face image to create the four subfeature templates. Each subfeature template has a mask applied, then the template pyramid is created from these masked templates, and finally the template pyramids are saved to the database Visage. Manifest records for the template pyramids are added to the Visage manifest. Facets provides several tools and dialogs to assist with this process.

---

[14] Potentially correct faces have enough correctly detected subfeatures to create a face estimate that includes all subfeatures, although possibly some detected subfeatures are incorrect.

[15] These subfeature templates are the initial templates. In our test they were always the left eye.

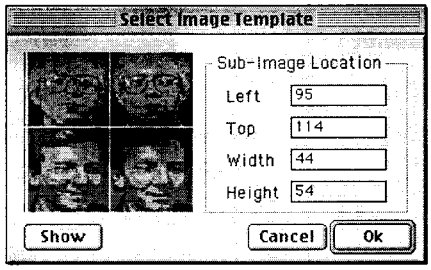

Figure 11 - Selecting a face to create a template.

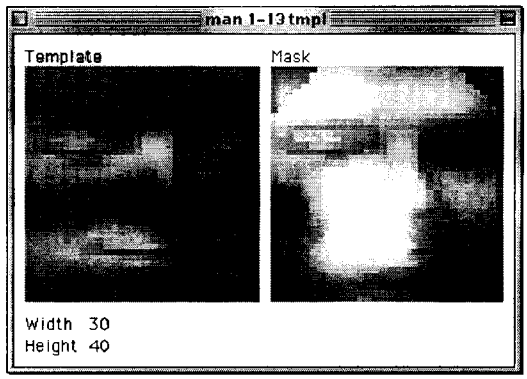

Figure 12 - Editing the template mask.

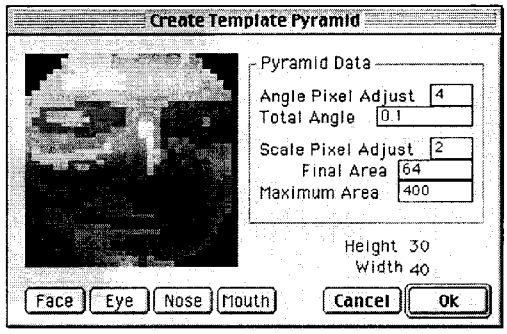

Figure 13 - Creating a template pyramid from a template.

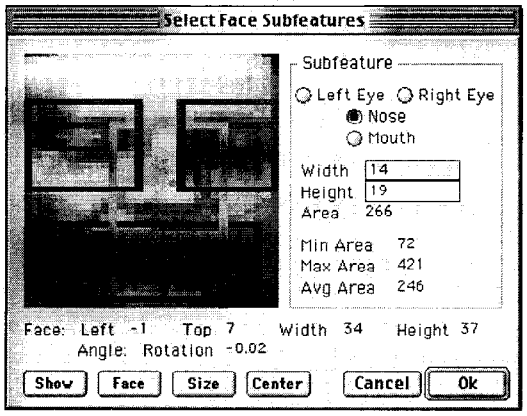

Figure 14 - Selecting subfeatures from a face image to create a template.

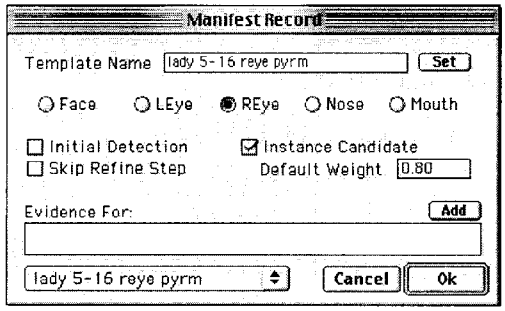

Figure 15 - Creating a manifest record for a template pyramid.

Detection accuracy was determined manually by inspection. Accurate detection for simple faces required that the face outline contained the eyes, nose, and mouth in the image. The maximum face size was judged more subjectively for reasonable appearance. Accurate detection for compound faces required that all subfeature outlines contain their respective subfeature, and not contain any other subfeature, in the image. Some small overlap was permitted. Detected faces deemed accurate are specified as correct, otherwise failed. Subjectively, the compound

templates produced closer matches to the actual face. Some examples are visible by comparing Figure 16 and Figure 17.

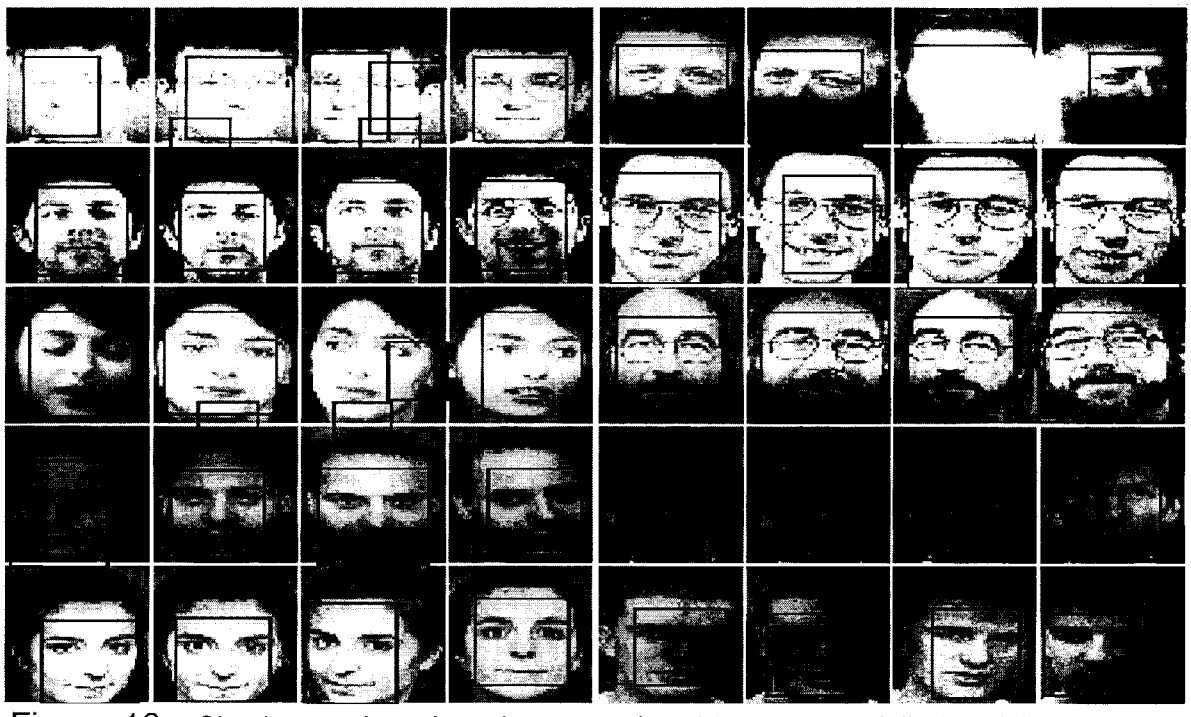

Figure 16 - Simple template detection example. 39 correct, 7 failed (6 false-positives, 1 false-negative).

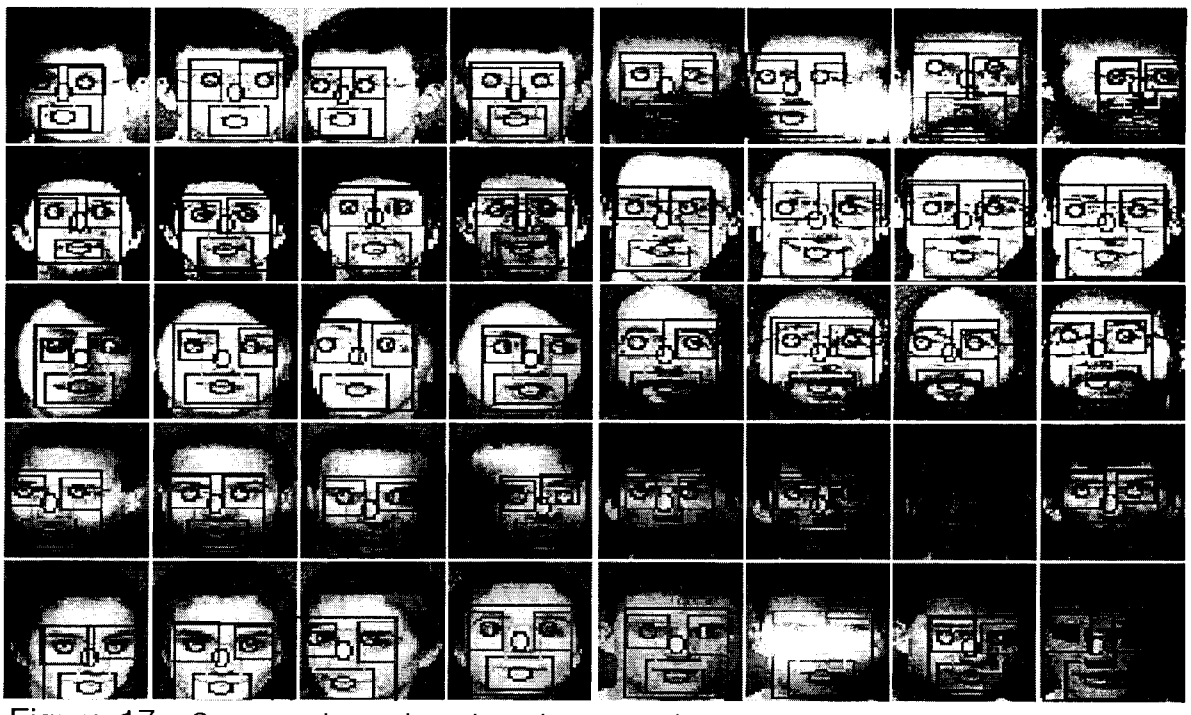

Figure 17 - Compound template detection example. 40 correct, 0 failed.

The simple template technique required 54 templates, totalling 2,010 KB in size, to cover the training set with 100% accuracy. By comparison, the compound template technique required 22 left eye templates (the initial detection templates), 44 right eyes, 25 noses, and 74 mouths, for a total of 165 templates, totalling 2,604 KB in size[16].

Tests were performed on a 400Mhz PowerPC based PowerMac G4 with 128MB. The Velocity Engine was not used. Times are in system ticks (1/60th of a second).

3.2 Training Performance

The performance characteristics of the simple and compound techniques were compared during the template acquisition process. The key result of these tests indicate how domain coverage, detection times, and resource requirements increase as templates are added to the database.

For each new initial template, timing and accuracy data were collected from the detection run. For compound templates, the detection run for initial template N, was performed just before adding initial template N+1. This ensures that as many complete faces as possible are detected from the initial evidence provided by initial template N.

The following accuracy tests show the domain coverage and accuracy as a function of the number of initial templates. The sharp increase in coverage and accuracy seen near the end, in Figure 19 below, is because incomplete compound faces (i.e., with four subfeatures, at least one subfeature correct, but not all correct) were completed (i.e., correctly matching subfeature templates were added) during the final step of the template acquisition process. Compound templates showed better accuracy throughout. Simple templates made 36 false positive identifications for the final run, while compound templates made only 1 false positive identification. Almost all failed matches for compound templates were incomplete faces, as compared to the simple templates where most failures were non-faces. One potential explanation is that compound templates are less likely to detect smaller features (this issue is addressed further in the other tests below).

---

16 Why the left eye space was covered with 22 templates, and the right eye space represented by twice that (44 templates), is not clear. It may result from biases in the template acquisition process.

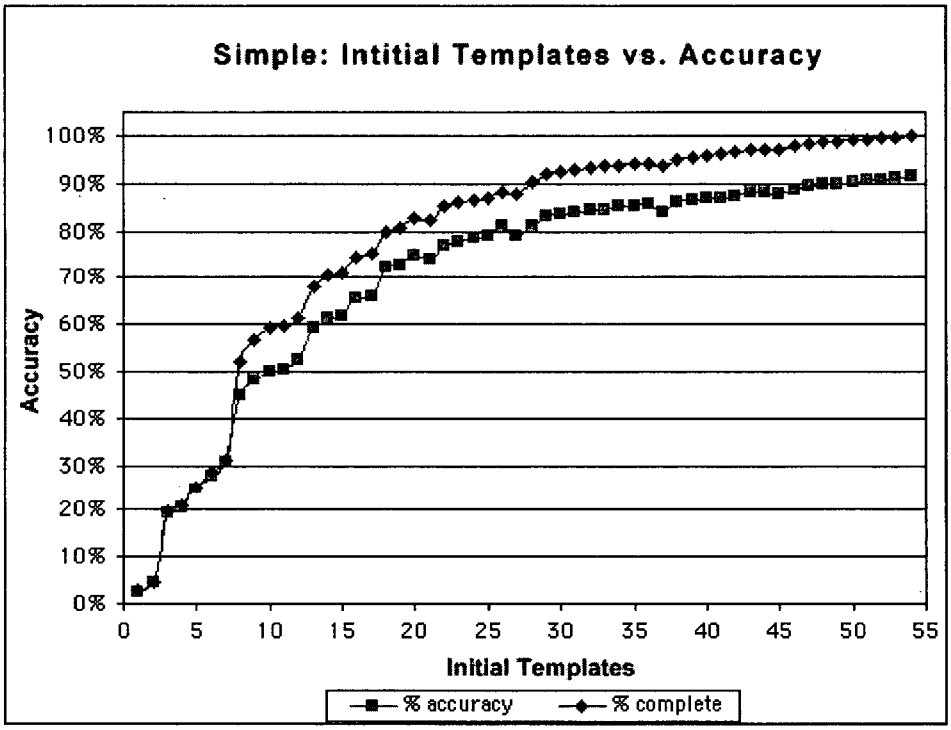

Figure 18 - Simple template accuracy test. Completeness denotes true-positive matches only, while accuracy accounts for false matches.

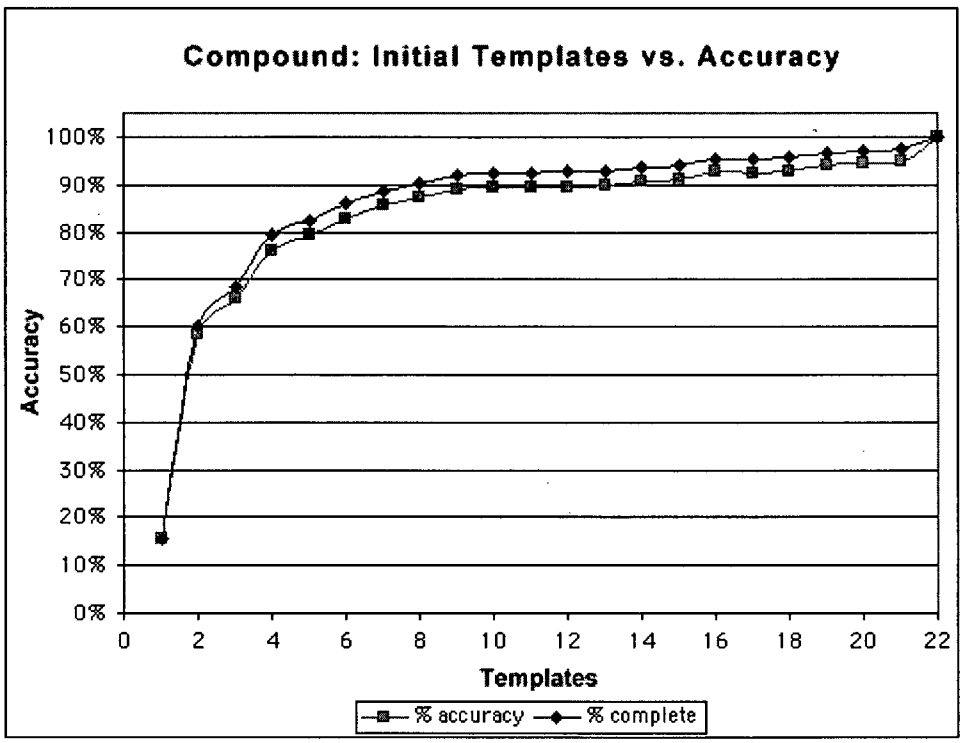

Figure 19 - Compound template accuracy test. Completeness denotes true-positive matches only, while accuracy accounts for false matches.

Compound templates also demonstrate better detection rates from available templates; however, this may result from the larger number of templates initially in the compound template database (i.e., more flexibility of the compound template to represent faces with any appropriate combination of subfeatures, it may also be that the larger number of templates cover a larger portion of the face domain). The growth of the template database is discussed later. In Table 1 the completeness of the database is shown compared to the percentage of correctly detected faces. The completeness is shown using initial templates, and additionally for the compound database, the total number of templates in the database. For compound templates, the initial templates are the constraint on the number of faces it is possible to detect (since initial templates are used in the intial bottom-up search). For simple templates, the initial templates are also constraints of the number of faces it is possible to detect (since they are the faces).

Table 1 - Domain detection coverage compared to template database completeness

Detected % faces detected	**Simple Templates** # initial templates / **(% database complete)**	**Compound Templates** # initial templates / **(% database complete)** / [# total template / (% complete total)]
60%	10 (**19%**)	2 (**9%**) [85 (52%)]
70%	14 (**26%**)	3 (**14%**) [97 (59%)]
80%	18 (**33%**)	4 (**18%**) [108 (65%)]
90%	28 (**52%**)	8 (**36%**) [130 (79%)]
95%	39 (**72%**)	16 (**73%**) [143 (87%)]

As the template database increases in size, the detection times for simple templates increases linearly, while the increase is sub-linear for compound templates[17].

[17] This comparison is complicated by different nature of the two approaches. It is not clear if growth of the database is better counted as initial templates, total template, size, or coverage of the training face domain. Although initial templates for simple templates represent the entire database, and for compound templates only a fraction thereof; they do represent an important fraction. Initial template in the compound approach have the greatest influence on detection times.

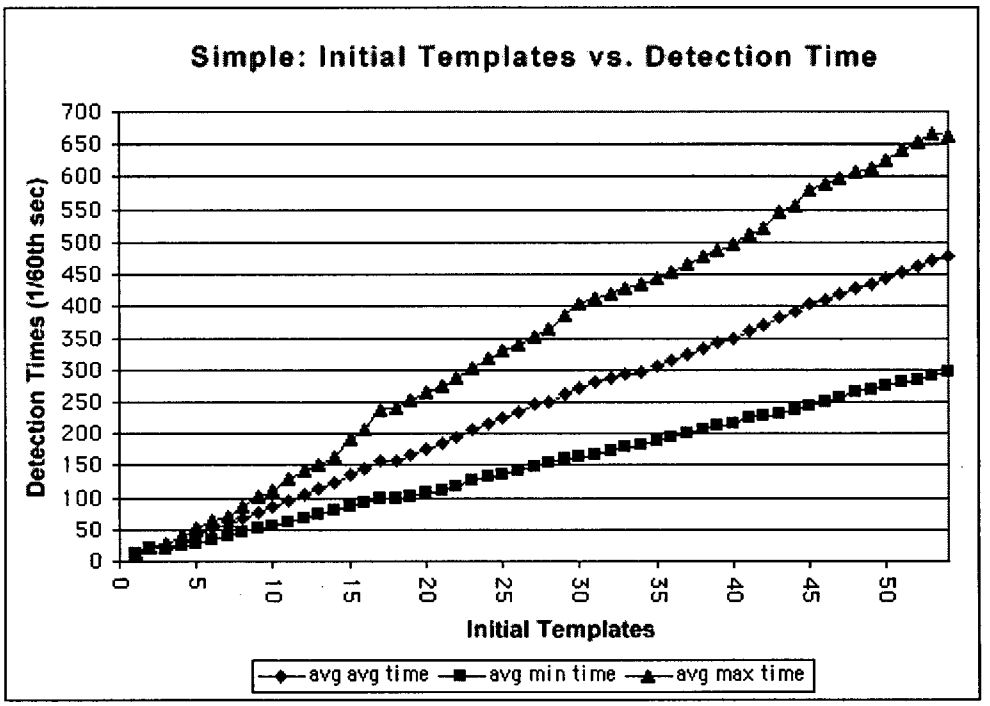

Figure 20 - Simple template detection times.

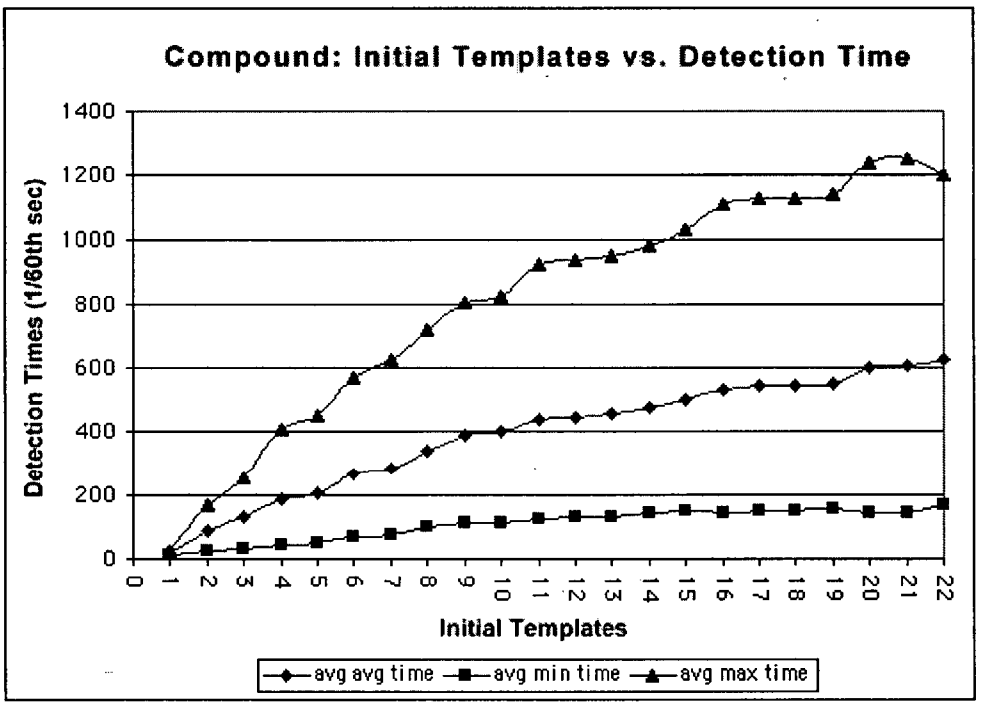

Figure 21 - Compound template detection times.

Average detection times for compound templates were nearly twice as long as those for simple templates. One reason may be the large number of feature instances detected. On the final detection run, simple templates detected a total of 2,201 faces, while the compound approach

produced 3,510 faces, 7,543 left eyes, 7,933 right eyes, 4,815 noses, and 12,415 mouths, for a total 36,216 features. However, compound templates produced consistently faster times to detect the first face, and the search optimizations possible with compound templates demonstrate their effectiveness when the average and maximum detection times for the best faces are compared to the corresponding detection times for all faces.

Table 2 - Detection times for final detection run.

	Simple Templates (1/60th second)	**Compound Templates** (1/60th second)
Minimum Time (**best faces**)	**296**	**169**
Average Time (**best faces**)	**478**	**624**
Maximum Time (**best faces**)	**664**	**1201**
Minimum Time (**all**)	292	65
Average Time (**all**)	517	1003
Maximum Time (**all**)	678	1728

The growth of the template databases during template acquisition was similar to the increase in detection times. The simple template database increased linearly from its initial 58 KB to its final 2,010 KB[18], passing the midway 1,005 KB size with 28 templates. The compound template database grows in size from the initial 458 KB, to the final 2,604 KB, passing the half way 1,302 KB size with just 3 initial templates (97 templates total). With 11 initial templates (136 templates total) the database is 2,117 KB in size.

3.3 Testing

The performance of both techniques was evaluated on the Nottingham and Yale image sets using the template database acquired previously from the training set. These image sets are described in chapter 1, section 1, above.

The simple template technique is more likely to discover smaller faces.

---

[18] The template database size includes index overhead.

To make the comparison as fair as possible, simple template matches below a given threshold were ignored. The threshold was determined from the smallest feature discovered by the compound template approach over the same data set. For the Nottingham image set, the threshold was 573 pixels in area. For the Yale image set, the threshold was 409 pixels in area. The results from this adjustment are noted below. Compound templates may find a face, but not find all four of the subfeatures correctly. An adjustment was made for face matches with three correct subfeatures, so that the incomplete face was counted as 3/4 of a face. This adjustment for compound results is also noted below.

Table 3 - Face detection results for the Nottingham and Yale image sets.

	Simple Templates % faces detected / (% **adjusted**)	**Compound Templates** % faces detected / (% **adjusted**)
Nottingham	65.63% (**69.58%**)	87.71% (**93.02%**)
Yale	15.56%, (**15.56%**)	31.11% (**35.14%**)
Both	34.45% (**35.94%**)	52.47% (**56.98%**)

The vastly better performance for the Nottingham image set is because it contained full frontal, expressionless faces, while the Yale image set contained frontal faces with a wide range of expressions. Therefore, training on different facial expressions would be necessary for reliable detection.

These tests suffered from high false positive rates. The detection accuracy is shown in Table 4 below.

Table 4 - Accuracy results for the Nottingham and Yale image sets.

	Simple Templates % accuracy / (% **adjusted**)	**Compound Templates** % accuracy / (% **adjusted**)
Nottingham	14.26% (**35.92%**)	53.43% (**57.96%**)
Yale	0.88% (**1.18%**)	11.29% (**13.25%**)
Both	5.93% (**14.28%**)	27.19% (**30.12%**)

The compound template technique performed better, sometimes substantially, in all these tests.

3.4 Non-Faces

Both techniques were tried on three non-face images, using the complete template databases, to test for false positive performance. The images are labelled by content and are shown below. As discussed previously, the simple template technique is more likely to detect small features. To equalize the test, simple template features were limited to those greater than 611 pixels, the area of the smallest face detected by compound templates.

Table 5 - Non-face image results.

	Simple Templates # false positive faces (# **adjusted false positives**) / # total faces	**Compound Templates** # **false positive faces** / # total faces
Tree (399x536)	61 (**19**) / 365	**18** / 398
Sky (350x241)	21 (**11**) / 63	**5** / 157
Canyon (670x450)	47 (**14**) / 118	**8** / 416

The post-detection test images for Tree, Sky, and Canyon are shown below:

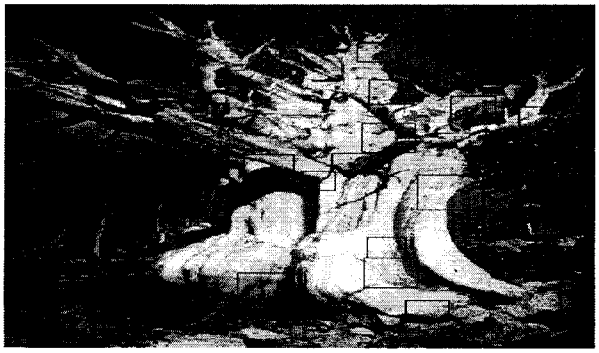

Figure 22 - Tree with 19 false positives using simple templates.

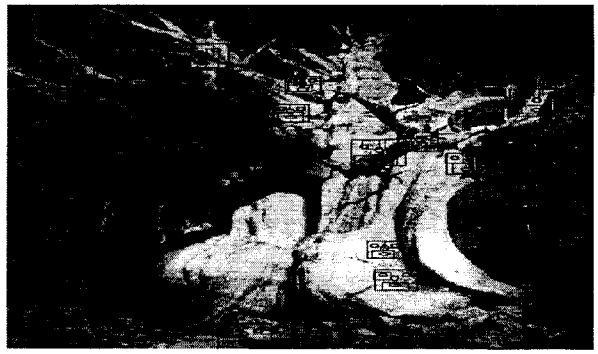

Figure 23 - Tree with 18 false positives using compound templates.

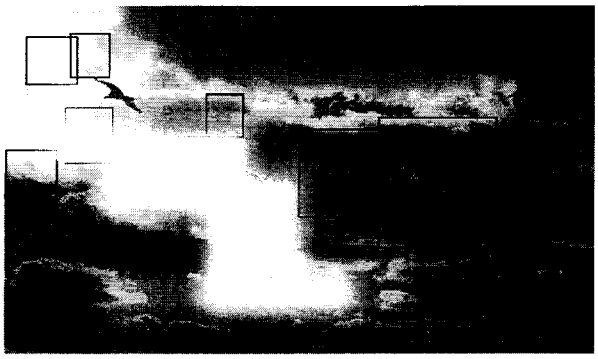

Figure 24 - Sky with 11 false positives using simple templates.

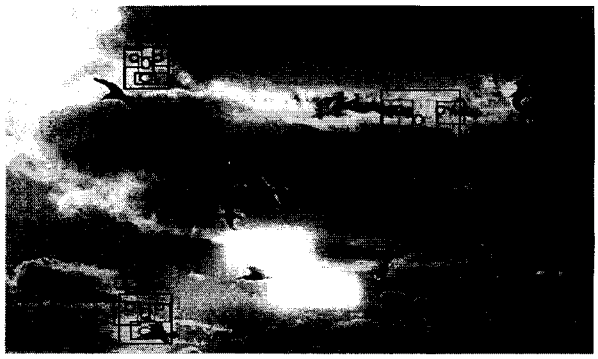

Figure 25 - Sky with 5 false positives using compound templates.

Figure 26 - Canyon with 14 false positives using simple templates.

Figure 27 - Canyon with 8 false positives using compound templates.

Again, as in the previous Testing subsection, the compound template technique provides better false positive discrimination.

The relative detection times for compound templates also improve significantly for these test images:

Table 6 - Non-face detection time results.

	Simple Template # minimum time / **# average time /** # maximum time	**Compound Templates** # minimum time / **# average time /** # maximum time
Tree (399x536)	1289 / **2090** / 2387	1147 / **1776** / 2668
Sky (350x241)	538 / **696** / 872	610 / **922** / 1355
Canyon (670x450)	1255 / **3079** / 3628	1646 / **2332** / 3522

This indicates that compound templates are dependent on the number of features, more than on image area alone. This result is explored below in the density tests.

3.5 Scale and Density

The performance of simple and compound templates was compared over images with varying scale and facial density. The test image was a 40 face subset from the training set (shown in Figures 16 and 17). For the scaling tests, 9 scaled images were produced at 50%, 75%, 85%, 95%, 100%, 125%, 150%, 175%, and 200% the size of the original image. For the facial density tests, 5 images were created by covering some of the faces with a textured swatch, producing images with 40, 30, 20, 10, and 2 faces visible (corresponding approximately to 100%, 75%, 50%, 25%, and 5% of the image uncovered).

For the detection accuracy tests, an adjusted performance value is calculated based upon the smallest face detected in the compound test on the same image. For the simple tests this greatly improves the accuracy and correctly detected face counts for the large images; but it hinders the performance for the smallest image. The reason is that many small false positive faces detected by simple templates in the large images were removed, while in the small image, the compound technique did very poorly and was unable to detect the small features, so the threshold was too large and the small features removed from the simple template search were actual faces.

The simple template approach performed much better overall producing a gradual face detection performance decay away from the nominal 100% case. The one bright spot for the compound template test is that, unlike the simple templates, it did perform with 100% accuracy in the 100% image size case. These results may be the result of an implementation issue.

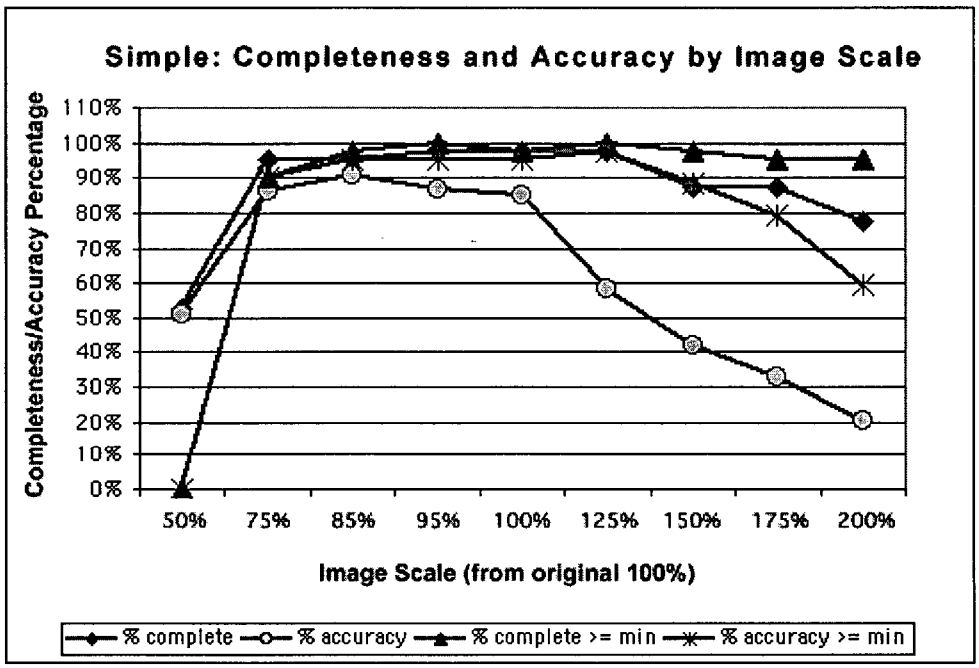

Figure 28 - Simple template detection performance by image scale.

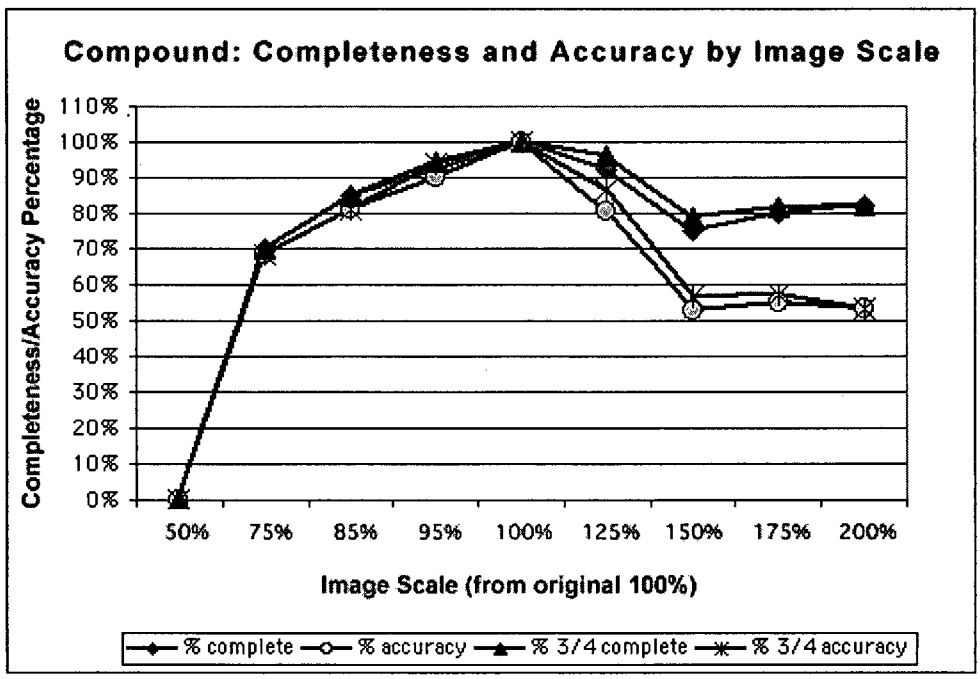

Figure 29 - Compound template detection performance by image scale.

Detection time performance was also measured over the different images scales. The graphs in Figures 30 and 31 indicate linear growth for both simple and compound templates. The slope of the average detection time line is 9.8 ticks per 1000 pixels for simple templates, and 14.1 ticks per 1000 pixels for compound templates.

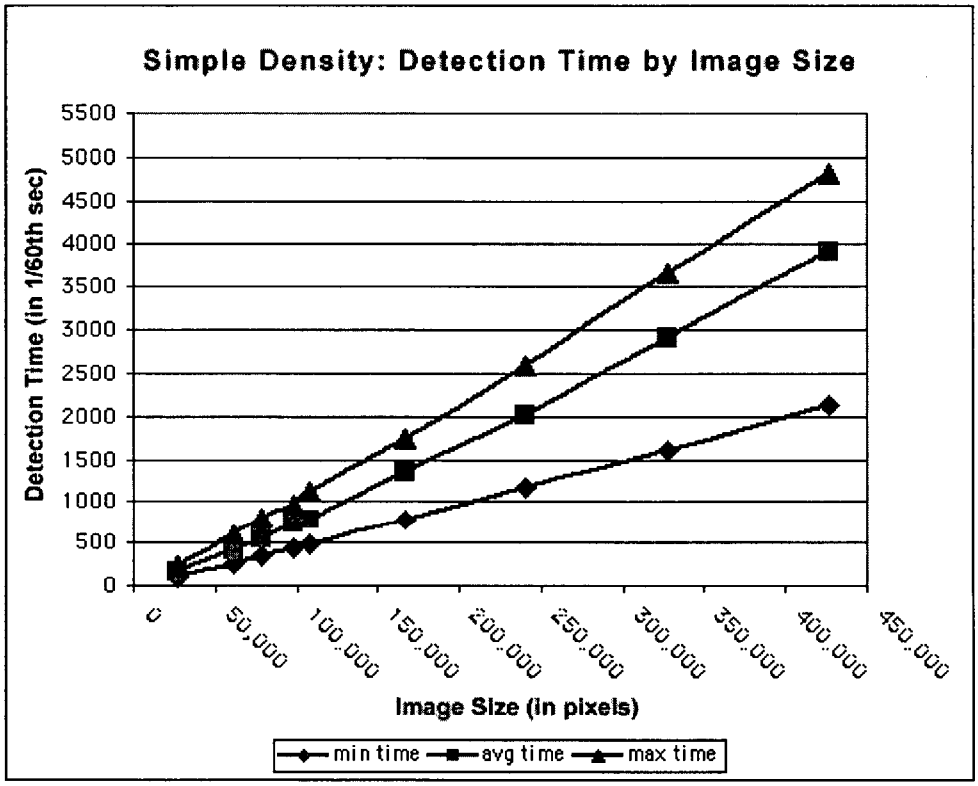

Figure 30 - Simple template detection times by image scale.

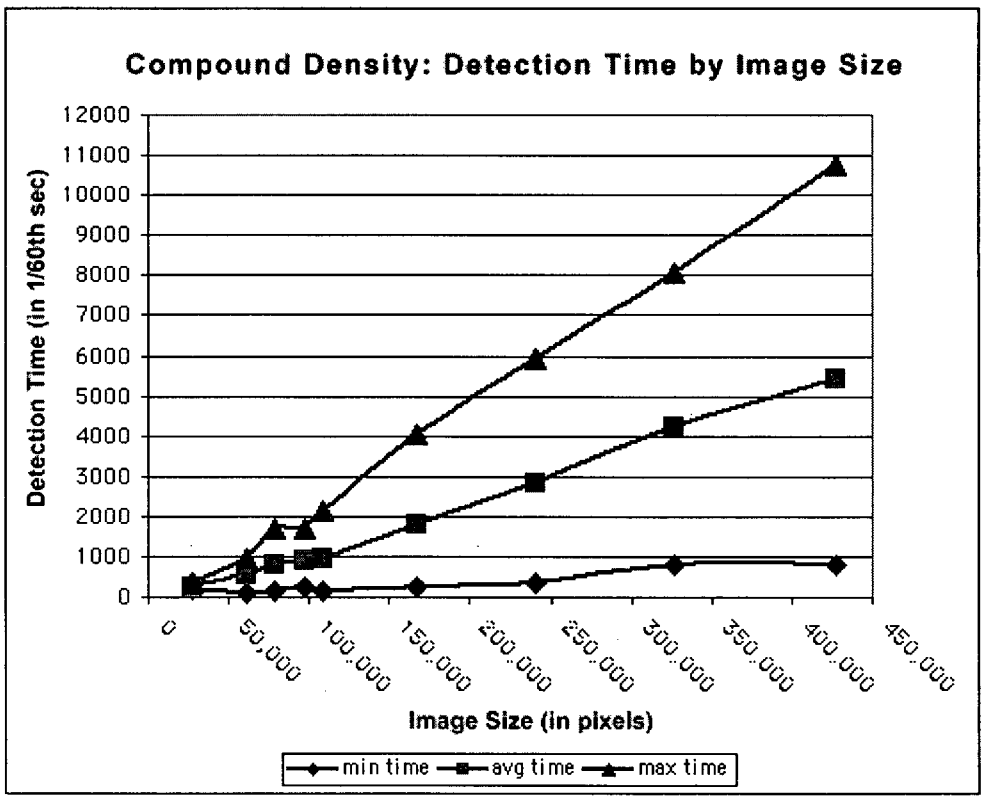

Figure 31 - Compound template detection times by image scale.

The detection times for simple templates depends largely on the image size, and very little upon its contents. On the other hand, detection times for the compound template approach depends on the number of unique initial features discovered, which does depend on the image contents. There is always the overhead of the initial bottom-up search which does depend on image size. The graphs in Figure 32 and 33 illustrate this.

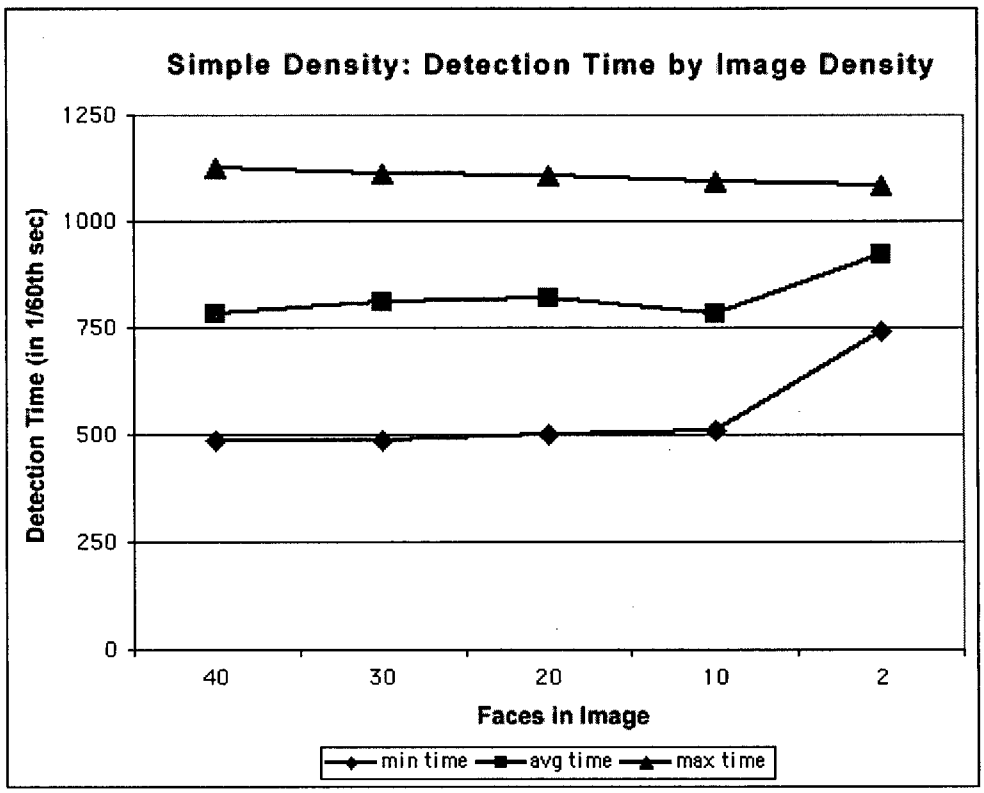

Figure 32 - Simple template detection times by facial density.

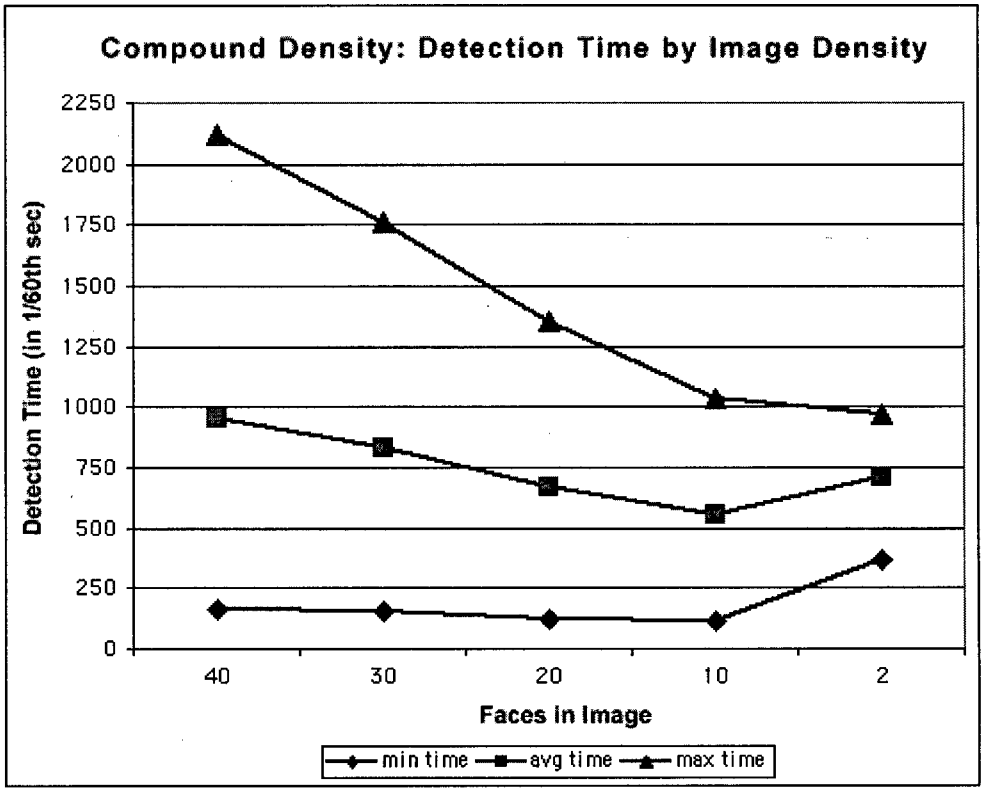

Figure 33 - Compound template detection times by facial density.

3.6 Comparisons

Comparative tests were performed to determine the effectiveness of combining the bottom-up and top-down search, and of the aggregate search optimization. The test image was the same 40 face image, at 100%, used for the scale and density tests. The five tests performed were of the following type:

Simple	Searching the entire simple template database over the entire test image (all 54 templates). This amounts to the standard simple templates.
Compound	Searching the entire compound template database (all 165 templates) over the entire test image.
Primary	Search only the initial detection templates (22 templates) over the entire test image (i.e., only perform the bottom-up search).
Standard	The standard compound templates technique as used throughout the testing process.

Unoptimized The aggregate search optimizations are not performed, instead search proceeds for all subfeatures over the localized feature space (combined bottom-up and top-down searches).

The results are shown in the Figure 34 graph below, which shows for each test the maximum, average, and minimum times for finding the best faces.

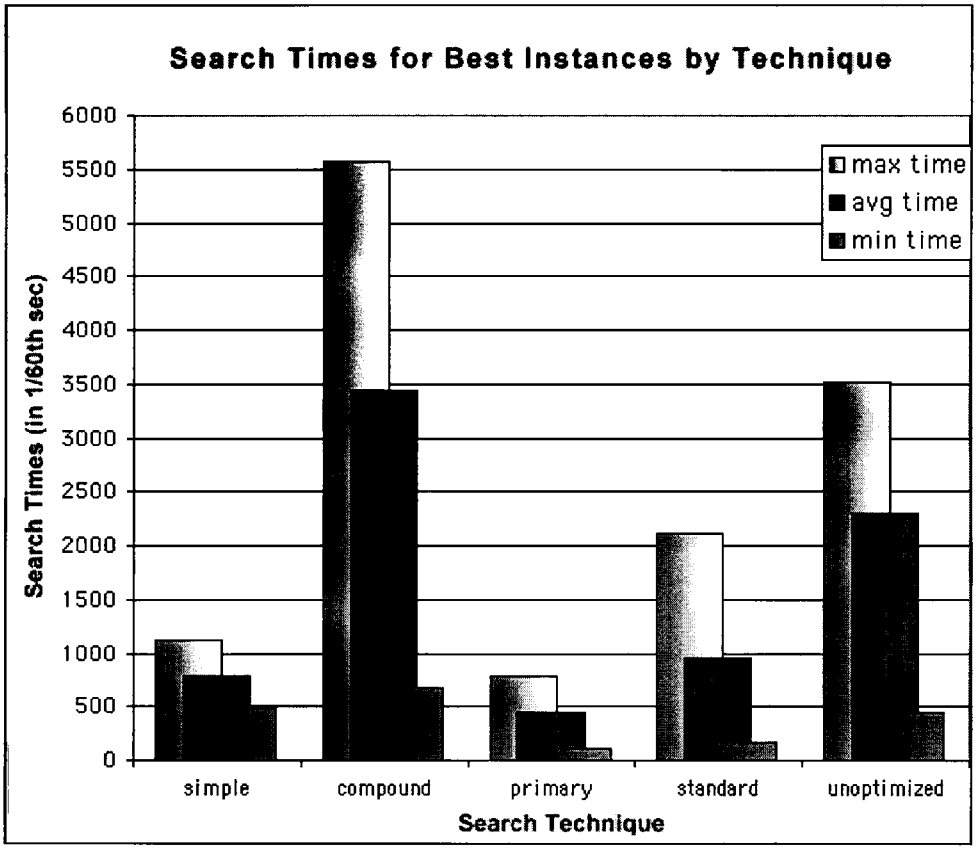

Figure 34 - Comparison of search time for various partial approaches.

The compound test essentially performs simple template detection for 165 templates. Although the number of templates is 3 times the number of templates used for the simple test, the detection times are nearly 4.5 times longer, even though the subfeature templates are smaller. This is partly due to the slightly higher relative resolution of the subfeature templates compared to the face templates (see Figure 8 for a visual comparison), the computational overhead required to prepare each template for search, and the higher probability that subfeature templates will match during the candidate search, thus requiring refine and instance searches too. Comparing the compound and unoptimized results shows

that combining bottom-up and top-down search is more efficient than bottom-up search alone. Comparing the standard results to the unoptimized results shows that optimizing the top-down search is possible and effective.

4 Conclusion

This paper described the background, motivation, techniques, and implementation behind the Facets face detection system. Test data was collected from this system comparing the two object recognition approaches: simple templates and compound templates. The results, while not conclusive, do show that the compound template approach has merits worth further exploration. In particular, compound templates provides better feature generalization, better detection times for feature sparse images, and slower detection time growths as new objects are added to the domain.

Heightened expectations for vision systems applications, such as real-world robotics, are sure to increase the size and complexity for object recognition domains. Complete usage of available knowledge, both of the domain and of the image, appears a plausible way to make the detection process more efficient. Combining bottom-up and top-down search processes is one way to use partial image knowledge (features found during the bottom-up search) and domain knowledge (the modeled spatial relations localizing the top-down search). A compound technique was proposed for combining the bottom-up and top-down search. The face domain was chosen as a tractable, extensible, and interesting domain for study. Existing research in face detection is extensive and provides for a variety of approaches. Sung and Poggio's face detection system utilizes top-down search with whole faces [Sung]. The top-down, whole faced, simple template technique was used as a comparative metric for the compound template approach. Other systems in a variety of domains [Matsuyama], [Shakungaga], use combined bottom-up and top-down search; however, the emphasis appears on the detection performance of the whole system, not efficiency. Facets implements a novel variation of a compound technique, and provides a testing platform to evaluate both detection and efficiency performance.

The simple template implementation used correlation between grey scale images and templates. Template searches were performed on the target image pyramid using a template pyramid and a three phase search. The compound template implementation used a 2D face model to represent facial spatial relations, four subfeatures, and simple templates to search for these subfeatures. Searching for subfeatures occurred during the initial bottom-up search, and during localized top-down search. The top-

down search was directed by face hypotheses generated from the face model. The search requests were scheduled based upon existing subfeature evidence, and further optimized by combining all subfeature searches for a single hypothesis.

Test results ranged from favourable, to inconclusive, to poor for compound templates. Compound templates appear to represent a larger face space with fewer features, possibly because of the combinatorial way that compound templates combine subfeatures. This is also evident in the Nottingham and Yale image tests, where compound templates detected 60% more faces. The detection times for compound templates also seemed to increase more slowly as the template database increased in size, when compared to the simple template tests; possibly because hypothesis management could increasingly reduce search duplication. This result could indicate that compound templates are more efficient for the larger databases needed for larger domains; however, this result is still open to interpretation depending on how the database growth is specified. Compound templates are more demanding of memory and computational resources in the chosen face domain, resulting in poorer absolute detection times than simple templates. Compound templates are well suited to image domains that are feature sparse, beating the simple approach in detection times for images with 50% face density or less.

Although the purely top-down approach often had faster detection times, the combination of bottom-up and top-down search was 150% faster then searching for all the subfeatures in a bottom-up manner. 350% faster when aggregate search optimization was used. While simple templates proved fast, they were not as accurate as compound templates; sometimes resulting in accuracy that was 2 to 10 times lower than compound templates. This might be surprising, considering that the full face template contained more of the face that the combined eye, nose, and mouth subfeatures combined. One possible reason is that the whole face templates were slightly smaller in relative size compared to the subfeature templates (i.e., a nose subfeature template would be slightly larger than the nose in the face template). Comparisons using equivalent resolutions would have been preferable, but must be saved for future experiments.

The implemented compound templates technique is one instance of the general compound and combined approach described previously in chapter

1, section 3. The approach is extensible to other domains, to compound models with higher number of levels, and to any combination of underlying image analysis techniques. One future direction for development would create a more generalized implementation and evaluate this in a larger domain such as articulated objects, or human bodies [Felzenszwalb]. The approach would also benefit from more formal models, such as a Bayesian approach [Viola], or formal techniques to calculate facial pose [Shakunaga]. Another possible improvement would replace the underlying simple template detection with more robust techniques, such as principle components [Sung], [Shakunaga], or combine it with other techniques, such as flexible templates [Lanitis], [Yuille].

5 References

Beymer, David J. Face Recognition Under Varying Pose from MIT AI Memo, no 1461, 1993

Bottoni, P. Cinque, L. Levialdi, S. Mussio, P. Matching the Resolution Level to Salient Image Features in Pattern Recognition, vol 31, no 1, pg. 89-104, 1998

Bräutigam, Carsten G. Eklundh, Jan-Olof. Christensen, Henrik I. A Model-Free Voting Approach for Integrating Multiple Cues from unknown, pg.735-750.

Brunelli, Roberto. Poggio, Tomaso. Face Recognition: Features versus Templates in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 15, no 10, pg. 1042-1052, 1993

Burt, Peter J. Smart Sensing within a Pyramid Vision Machine in Proceedings of the IEEE, vol 76, no 8, pg. 1006-1015, August 1988

Felzenszwalb, Pedro. Huttenlocher, Dan. Efficient Matching of Pictorial Structures. at http://www.ai.mit.edu/people/pff/blobrec/blobrec.html and http://www.ai.mit.edu/people/pff/blobrec/blobrec2.ps.gz

Hofstadter, Douglas. Fluid Concepts and Creative Analogies. Basic Books, 1994

Jeng, Shi-Hong. Liao, Hong. Han, Chin. Chern, Ming. Liu, Yao. Facial Feature Detection using Geometrical Face Model: An Efficient Approach in Pattern Recognition, vol 31, no 3, pg. 273-282, 1998

Konen, Wolfgang. Comparing Facial Line Drawings with Gray-Level Images: A Case Study on PHANTOMAS at http://www.zn.ruhr-uni-bochum.de

Lanitis, Andreas. Taylor, Chris J. Cootes, Timothy F. Automatic Interpretation and Coding of Face Images using Flexible Models in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 19, no 7, pg. 743-755, 1997

Matsuyama, Takashi. Hwang, Vincent Shang-Shouq. SIGMA: A Knowledge-Based Aerial Image Understanding System. Plenum Publishing Corporation, 199?

Milanese, Ruggero. Gil, Sylvia. Bost, Jean-Marc. Wechsler, Harry. Pun, Theirry. Integration of Bottom-Up and Top-Down Cues for Visual Attention Using Non-Linear Relaxation in Berkeley Technical Reports, vol 94, no 14, pg. 1-6, 1994

Moore, David S. McCabe, George P. Introduction to the Practice of Statistics, second edition, W. H. Freeman and Company

Rowley, Henry A. Baluja, Shumeet. Kanade, Takeo. Neural Network-Based Face Detection in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 20, no 1, pg.23-38, 1998

Shakunaga, Takeshi. Integration of Eigentemplate and Structure Matching for Automatic Facial Feature Detection in IEEE Journal - Unknown (0-8186-8344-9/98), pg. 94-99, 1998

Sung, Kah-Kay. Poggio, Tomaso. Example-Based Learning for View-Based Human Face Detection from MIT Lab Report, pg. 1-8

Takacs, Barnabas. Wechsler, Harry. Detection of Faces and Facial Landmarks using Iconic Filter Banks in Pattern Recognition, vol 30, no 10, pg. 1623-1636, 1997

Viola, Paul A. Complex Feature Recognition: A Bayesian Approach for Learning to Recognize Objects from MIT AI Memo, no 1591. 1996

Yuille, Alan L. Hallinan, Peter W. Cohen, David S. Feature Extraction from Faces Using

Deformable Templates in International Journal of Computer Vision, vol 8, no 2, pg. 99-111, 1992

Weng, John J. Hwang, Wey-Shiuan. Toward Automation of Learning: The State Self-Organization Problem for a Face Recognizer in IEEE Journal - Unknown (0-8186-8344-9/98), pg. 384-389, 1998