Probabilistic Hierarchical Image Models

by

Ya Jin

B.S., Wuhan University, 1998
M.S., Oklahoma State University, 2001
M.A., Brown University, 2004
M.S., Brown University, 2005

Thesis

Submitted in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy
in the Division of Applied Mathematics at Brown University

Ph.D. Advisor: Stuart Geman

PROVIDENCE, RHODE ISLAND

May 2006

This thesis by Ya Jin is accepted in its present form by the
Division of Applied Mathematics as satisfying the
thesis requirement for the degree of
Doctor of Philosophy

Date ....
Stuart Geman

Recommended to the Graduate Council

Date ....
Elie Bienenstock

Date ....
Basilis Gidas

Approved by the Graduate Council

Date ....
Sheila Bonde, Dean of the Graduate School

The Vita of Ya Jin

Ya Jin was born in the city of Xuzhou, Jiangsu province on July 6th, 1976.

He graduated from the First High School of Zhenjiang City. In 1994, he started his undergraduate study in Wuhan University and received his Bachelor of Science degree in Computer Science in 1998.

In 1999, he came to the Unites States and received Master of Science degree in Mathematics from Oklahoma State University two years later. Since 2001, he has been attending the Ph.D. program in the Division of Applied Mathematics at Brown University. During the Ph.D. program, he received a Master of Arts degree in Economics in 2004 and a Master of Science degree in Applied Mathematics in 2005 at Brown University. This dissertation was defended in January 27, 2006.

Abstract of "Probabilistic Hierarchical Image Models," by Ya Jin, Ph.D., Brown University, May 2006

This thesis studies the probabilistic modeling of images. Two topics are covered. One is about the theoretical justification for efficient image representation; the other is about building a compositional machine to interpret real images.

Chapter 1 overviews the major methodologies developed in the field of computer vision and chapter 2 briefly introduces the methodology we adopt.

Chapter 3 focuses on object/scene representation. It explores theoretical supporting evidence for the hierarchical representation against holistic representation through a statistical testing argument. In conclusion, compositional system, with a built-in hierarchical structure, has provable better performance.

Chapter 4 focuses on implementation of a scene interpretation system (with application to license detection), which is capable of parsing a scene or detecting uncertain number of targets with cluttered background accurately. Under the Bayesian framework, all the components (i.e., prior model, data model and posterior model) and their implementations are discussed in detail. Demonstration of the system spells out the important aspects of our methodology; it also provides empirical supporting evidence, which is consistent with the theoretical one covered in chapter 3.

Chapter 5 makes a conclusion and touches on some future directions.

Acknowledgments

I'd like to thank all the people who have helped me to complete my thesis.

Most of all, I want to express my deepest gratitude to my advisor, Professor Stuart Geman, for all the guidance and support, leading to the completion of my thesis. His inspiring ideas and valuable suggestions shape my thought and his generous help in all aspects benefits my career. This thesis would not be possible without his persistent help. Spending the last few years doing research under his guidance is the best investment I have ever made.

I want to thank my other committee members, Professor Elie Bienenstock and Professor Basilis Gidas, for reading my thesis, discussing it with me and giving me precious comments.

I appreciate greatly all the help from Professor William Jaco in Oklahoma State University, who introduced me to Brown University. Without his effort, I cannot possibly come here to pursuit my Ph.D. degree. I also want to give my best regards to Professor Igor Pritsker in Oklahoma State University, who was my advisor for my Master's thesis.

I want to acknowledge Professor Hui Wang, Matthew Harrison and Ting-Li Chen, who have been very helpful during my research. Also I want to thank my officemates Yanchun Wu, Wei Zhang and all the people in the Division for helping me in every aspect. I am always thankful to my parents, who support me spiritually all the time.

Contents

List of Figures

Chapter 1

Introduction

1.1 General Review

Recognizing materials, objects and scenes is one of the most useful functions of vision. Understanding this ability in humans, and reproducing it in machines is still a huge challenge. The main reason is that the same object of interest can demonstrate a huge variety of appearances under affine transformation (such as scaling, rotation), non-affine transformation (such as face expression) and other conditions (such as illumination, occlusion). The surrounding background is even more unpredictable; it could be highly structured clutter which shares the same feature or statistics as the foreground object.

During the last few decades, many theoretical and algorithmic frameworks have been explored with application to different kinds of visual tasks. In the theme of object detection, the majority of the proposed frameworks fall more or less into one of the two categories.

Discriminative Modeling

Discriminative modeling is very popular in classification related tasks. Based on an objective function of a set of feature statistics and their class-dependent distributions, discriminant methods classify an object by maximizing the objective function over classes. Supported by the theory of nonparametric inference, many elegant approaches have been proposed under this framework, for example, boosting [13], neutral network [6], support vector machine [35], decision tree [9]. To avoid the problem of over-fitting, regularization terms are explicitly put into some of the objective functions or practical methodologies such as cross validation [32], VC dimensionality [34], and measures of complexity [29] are applied. These remarkable tools, coupled with advances in computing technology, yield state-of-the-art performance in some applications, like face detection and hand-written digits recognition. It follows that if any vision task can be formulated as a classification problem, then by statistical learning theory, it is solvable as long as enough independent and identically distributed (i.i.d) training examples are provided. Unfortunately, a task like scene interpretation is by no means a traditional classification problem; the number of classes is essentially infinite. Involved feature can be any function of the pixel values; learning its true conditional distribution requires massive training examples. In real situations, we never have the luxury to have "enough" data. How can we suppose to parse a new natural image using classifiers trained over a small set of distinct natural images? We believe pure learning is infeasible and introducing some hidden global structure is the only outlet for such ambitious vision problems.

Generative Modeling

Models falling into the generative modeling framework include hidden variables accounting for the underlying structure of objects of interest, unlike the discriminative models that only involve directly measurable variables. Observed data are presumably generated by the generative model, even though they may rarely arise directly from the model. Most commonly used generative models can either be model-based (like mixture model, HMM, MRF, PCFG), or basis-based (like PCA, ICA, wavelet analysis). The former captures the regularity of objects of interest through detailed modeling, while the latter gives a good approximation of a whole image based on linear superposition of basis images. Bayesian inference is often applied in the model-based setting, where bias is conveniently introduced in choosing the model (structure) to represent the prior knowledge about the data, and inference is made on the hidden variables within the model, given new data. Recently, more effort has been devoted to generative hierarchical models, due to their representational flexibility and searching efficiency. Related works under this theme can be found in such places as Bayesian net [23], HMAX model [28], POP model [2], constellation model [27] and fragment-based approach [31]. Compared to the discriminative models, fewer training samples and better generalization results are achieved by the above models. Still there is a noticeable gap between their performance and human performance in challenging tasks (like finding and locating uncertain number of targets in an uncontrolled environment), due to the "invariance and selectivity" dilemma (for details see Chapter 3). Nevertheless, we do believe that certain types of hierarchical representations can provide a promising direction for decidedly improved performance.

1.2 Motivation

The more we understand ourselves, the better chance we have to devise intelligent machines to mimic human performance. To our knowledge of the brain (most at the cognitive science level, some at the neuroscience level), we can argue that in a good vision system, there has to be some necessary and indispensable components, which are intuitively compelling.

1.2.1 Hierarchy of Reusable Parts

Evidently, humans have the tendency to represent entities as hierarchies of reusable parts, which themselves are meaningful entities and can be reusable in near-infinite groupings of meaningful combinations to form new entities. Natural language is clearly the case; in fact, word morphology is the subject which studies the admissible combining rules to form words, not to mention the idea of hierarchy of reusable parts is the key to computational linguistics. In a visual world, physical objects and scenes naturally decompose into hierarchies of meaningful and generic parts, the direct supporting evidence is that humans developed and shared a collection of vocabulary containing semantic entities at different conceptual levels, which can be used to describe any scene at different detailed levels whenever needed. This phenomenon is the motivation for our work. Once our framework incorporates this idea, a few direct advantages are:

1. We may deal with different configurations of objects and clutter in a test image, without having to really have seen them in the training stage. In other words,

it introduces more freedom to accommodate a variety of presentations of both target and background. It also makes learning from a few examples possible.

2. Hierarchy of reusable parts introduces hierarchy of contextual constraints. In general, components can not be correctly interpreted without the contextual constraints imposed by their composition into a bigger one. Due to the fact that background clutter shares common parts with foreground objects all the time, this helps us to reduce false positives by identifying parts unambiguously. It is also a natural way to throw in the selectivity.

3. We don't need to learn everything from scratch. Given that we already learned the parts, learning an object is equivalent to only learning the arrangement between its composing parts. That is, by modeling the relationship between parts, we avoid the difficulty of modeling the joint distribution of parts. Without much trouble, we can compose models of parts into models of objects and models of objects into models of scenes in a recursive way.

Variations of this work have been done to explore different aspects of the general idea.

Feature Sharing. To realize the long-term goal of machine vision, i.e., a machine capable of detecting a large number of different objects under an uncontrolled enviroment from a small number of training images, feature sharing is inevitably involved. Feature sharing is a common subject in computer vision. In fact, application of features themselves like (simple) Gabor patch or (full-grown) SIFT feature [26] in any vision task already demonstrates their reusability in different representations of objects from scene to scene. Psychologically, Biederman [4] and others have argued that a small number of reusable parts, also called "geons," could produce a huge number of unique objects recognizable by humans, each of which is represented by a small number of parts in a certain spatial configuration. Biologically, the HMAX model [28], proposed by M. Riesenhuber and T.Poggio, is built on the primitive reusable parts, also called simple cells. Empricially, Krempp et al. [24] found that the growth rate of the number of distinct parts is only a logarithmic function of the number of learned class in a sequential learning task. Moreover, some current practical multiple-object recognition systems ([2],[33]) do start with primitive reusable features.

Context. Image segmentation based on simple low-level features can be very difficult without any high-level contextual knowledge. That is because in a purely bottom-up framework (without any top down influence), boundary detection is inherently ambiguous at the presence of image noise, shading or cluttered background. We need high-level hints like multi-scale cues [12] or class-specific information [8] to disambiguate the situation. Hierarchical models present a natural way to integrate multi-level contextual constraints ([14],[15],[19],[28]). These constraints can either be context free or context sensitive.

Efficient Representation. In his essay on Probability [25], Laplace mentioned that people more likely see "CONSTANTINOPLE" as a single word instead of random arrangement of 14 letters, suggesting that Bayesian principles are helpful in detecting meaningful groupings among many possible combinations equally favored by chance. Later, *suspicious coincidence* [3], proposed by Barlow, provides a useful principle for discovering meaningful groupings of "parts" into "objects" at different contextual levels; it implies that introducing a new label (representing co-occurrence of parts in certain constraints) can be efficient if the probability of joint appearance is much bigger than the product of its marginal distributions. Similarly, a detailed example is illustrated by Bienenstock et al. in [5], which reformulate these meaningful groupings as an efficient coding scheme. Bearing in mind all the similarities, (semantic) part/object hierarchical representation embodies Rissanen's MDL principle [29].

We will further explore this idea, especially the justification of the hierarchical structure, from the efficient discrimination point of view.

1.2.2 Coarse-to-fine Computation and Progressive Annotation

Coarse-to-fine computation refers to the strategy that least discriminant features are quickly tested at different positions during the early stage of scene analysis, and then based on the test results, computational resource is directed to more specific locations where more discriminant tests are needed. By postponing computationally intensive tests as long as possible, this scheme efficiently allocates the resources on the most interesting (ambiguous) region while ignoring homogenous regions with little structure. It is more or less like attention mechanism in biological vision systems. It is hard to imagine that any vision system capable of conducting scene analysis with uncontrolled background has nothing to do with coarse-to-fine computation. The idea of coarse-to-fine scheme can be implemented in different manners, like coarse-to-fine in scale [22] and coarse-to-fine in scope [1]. The scheme we will cover is coarse-to-fine in model. Models of primitive parts are coarser relative to the models of intermediate parts, meaning that a given simple part participates in many more possible interpretations than a given intermediate part. Similarly, models of intermediate parts are coarser relative to the models of objects. A general search framework for fast object detection will be touched upon. Since the main theme is scene interpretation instead of just detection-oriented tasks, we will mainly focus on a computation scheme that allows us to annotate a scene progressively. It is not natural to resolve the ambiguities all at once; instead during the search for the optimal interpretation, coarse and likely flawed interpretations emerge at the early stages of process, followed by finer and more accurate ones. Biological vision seems to work as an endless process.

1.3 Outline of the Thesis

There are two main foci in this thesis. The first is building a full-grown compositional machine under the Bayesian framework to conduct scene analysis for a real application, with the idea of a hierarchy of reusable parts explicitly encoded as a prior. The second is exploring some supporting (theoretical and computational) evidence to justify the hierarchical representation.

This work falls into the general framework for computational vision, notably known as "compositional vision," proposed by Geman, Bienenstock and their colleagues ([5],[19]). Some preliminary computational experiment can be found in Huang [22] and some learning related work can be found in Harrison [21]. Early related work mainly involves syntactic methods as seen in Fu [14], Grenander [20], originally introduced by Chomsky [10] in modeling language.

The thesis is organized in the following way.

Chapter 2 gives an overview of the key elements of a compositional system, that is, the prior model, the conditional data model and sampling from the posterior model.

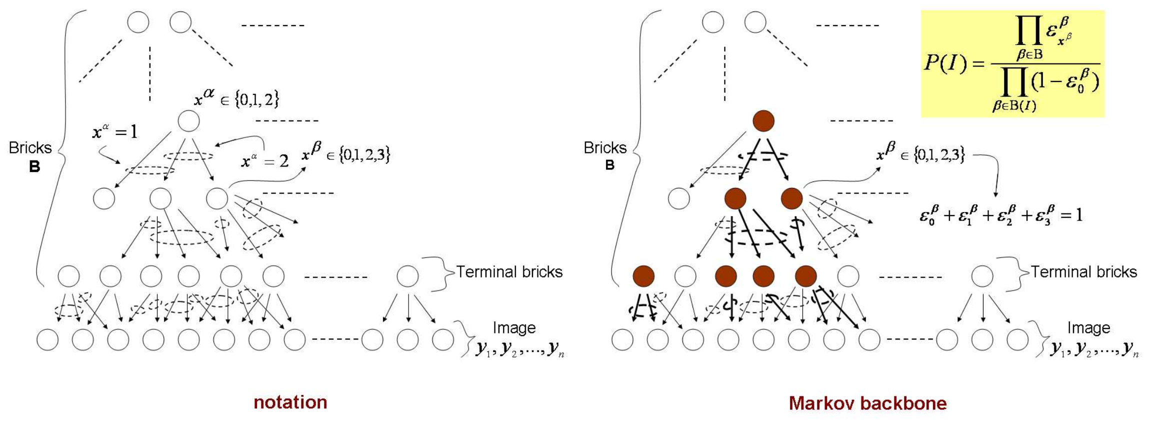
Chapter 3 states the theoretical results supporting the parts-based model over the whole model, which implicitly justify the hierarchical representation. We formulate it as a statistical testing argument (i.e., sequential testing versus whole testing) and link the proof directly with their ROC curves.

Chapter 4 is devoted to designing elements of a compositional machine mentioned in chapter 2 and illustrating the idea with a license plate demostration system. In such a system, a hierarchical representation of reusable parts is manually designed (hardwired), under which a coarse-to-fine searching engine is built to illustrate efficient computation.

Chapter 5 makes a conclusion and states some future directions.

Chapter 2

Overview of the Model



Bricks
B
$x^{\alpha} \in \{0,1,2\}$
$x^{\alpha} = 1$
$x^{\alpha} = 2$
$x^{\beta} \in \{0,1,2,3\}$
Terminal bricks
Image
$y_1, y_2, \ldots, y_n$
notation
$P(I) = \frac{\prod_{\beta \in B} \varepsilon^{\beta}_{x^{\beta}}}{\prod_{\beta \in B(I)} (1-\varepsilon^{\beta}_0)}$
$\varepsilon^{\beta}_0 + \varepsilon^{\beta}_1 + \varepsilon^{\beta}_2 + \varepsilon^{\beta}_3 = 1$
Markov backbone

Figure 2.1: Notation of a compositional system and its Markov backbone

The model is under the Bayesian framework. In this chapter, we briefly describe its components, that is, the prior distribution, the conditional data distribution and sampling from the posterior distribution (i.e., parsing scene). At the end, we will show that searching for the optimal parse is provably NP-hard, even for a toy example.

2.1 Composition System

The formulation is best motivated in two steps. In the first step, we construct a generative hierarchical model with built-in Markov property on a directed acyclic graph (see Figure 2.1). We call this 'Markov Backbone,' which serves as a reference distribution. It is then extended to 'compositional system' by introducing a non-Markovian term through a "perturbation" argument in the second step.

2.1.1 Markov Backbone

The whole graphical model involves layers of two-dimensional sheets of nodes, the one in Figure 2.1 only shows a vertical slice of it, for illustration purposes. The nodes on the bottom line represent a slice of an observable two-dimensional image pixel array. All the above nodes are hidden variables and a legitimate configuration (definition see below) of all the hidden nodes defines the system's internal state. Due to their reusability across legitimate configurations, we call them "bricks." The nodes right above the image pixels are called "terminal bricks."

Bricks represent semantic entities, like edge, L-junction, shape, parts and objects. Precisely, terminal bricks are associated with local primitive features, top level bricks are associated with objects of interest and bricks in the intermediate layers are related to parts with intermediate complexity. The specific assignment is application dependent. In a license detection application, one assignment would be as in Figure 2.2. Note in this case, each sheet of bricks except the top layer is interspersed with a few different types of bricks. For example, on the fourth layer above the image pixels, 2-letter bricks, 3-digit bricks, 3-letter bricks, 4-digit bricks and plate boundary bricks are interspersed evenly across the sheet. The total number of involved bricks is above 500,000.

The state of a brick, say brick $\beta \in B$ (the set of all bricks), is a discrete random variable, $x^\beta \in 0, 1, ..., n^\beta$, where n^β is the number of active states for brick β. A probability vector $(\epsilon_0^\beta, \epsilon_1^\beta, , \epsilon_{n^\beta}^\beta)$ is assigned to its states and they sum to 1 (see Figure 2.1). When $x^\beta = 0$, brick β is off, meaning the entity represented by brick β is not present. When $x^\beta \in 1, 2, .., n^\beta$, brick β is on (meaning that the entity is present) and its particular active state refers to the way (out of a set of allowable ways) the entity is instantiated. In other words, for each brick β in the above set (i.e., set of non-terminal bricks), a collection of subsets of children bricks drawn from the layers below is hardwired to brick β, from which brick β selects one subset of children bricks whenever it is on. We refer to this as a "composition" of children (parts) bricks into a parent (object) brick. A legitimate configuration (also called interpretation) refers to a subgraph of on bricks, each instantiated by one allowed subset of children bricks which have to be on by themselves (see right panel of Figure 2.1). Such subgraphs are called complete and represent a semantic labeling of a scene.

Consider the space of interpretations (complete subgraphs); we can impose a probability distribution over it with the property that sampling from the distribution is equivalent to sampling directed acyclic graphs in a Markov fashion. This statement suggests a proper distribution and its justification at the same time. For each interpretation I,

$$P(I) = \frac{\prod_{\beta \in \mathbf{B}} (\epsilon^{\beta}_{x^{\beta}})}{\prod_{\beta \in \mathbf{B}(I)} (1 - \epsilon^{\beta}_{0})} \tag{2.1}$$

where $\mathbf{B}$ is the set of all bricks and $\mathbf{B}(I)$, the "below set," is the set of all *on* bricks that are not roots of the directed subgraph I.

To verify that $\sum_{I \in \mathcal{I}} P(I) = 1$, let's go through its sampling argument more carefully. First, choose the state of each top-layer brick independently according to its probability vector. Next, choose the state of each brick on the next-to-top layer independently according to its probability vector, except that those selected children of on bricks are conditioned themselves to be on. Continue downward until we are done with choosing states for terminal bricks in the same fashion. Based on this constrained sampling scheme, each sample is a legitimate configuration and the sample space is the space of interpretations. The weight of each sample under such generation mechanism is consistent with its probability as defined in Eqn 2.1. That leads to the conclusion that $P(I)$ is a true distribution over I and from the sampling scheme, the Markov property (i.e., given the state configuration of bricks on the intermediate layer, higher-layer bricks are independent of lower-layer bricks) with respect to the directed acyclic graphs (represented by arrows in Figure 2.1) is also established. The model in Eqn 2.1 is what we called the Markov backbone.

In the license application, given the semantic assignment of bricks (in Figure 2.2) and manually hardwired children sets (in §4.1), we can actually sample from the Markov backbone, and by looking at the samples we can appreciate the characteristics of the model. A 4-digit sample under the Markov backbone is shown in the left panel of Figure 2.3 (for sampling detail see §4.4.1), though it doesn't look like a regular 4-digit sample. From the image, we can see parts of each digit are not properly composed and the four digits are not well aligned. Other than that, everything seems

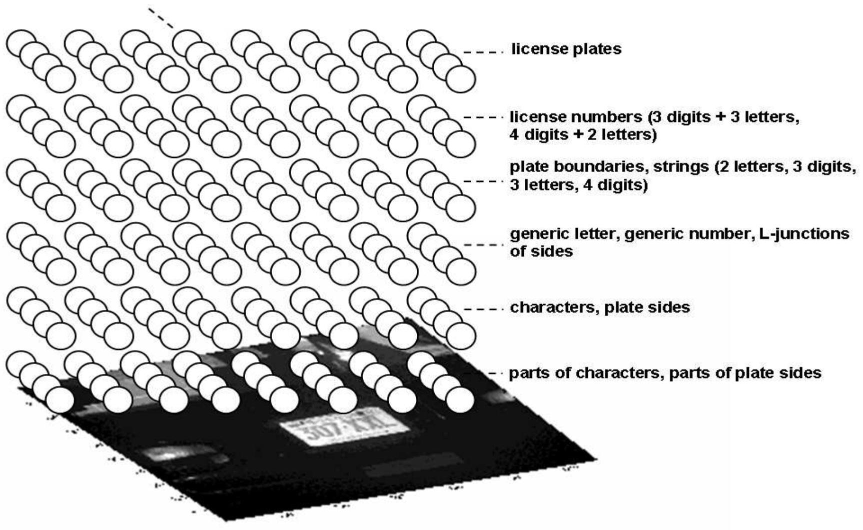

license plates
license numbers (3 digits + 3 letters, 4 digits + 2 letters)
plate boundaries, strings (2 letters, 3 digits, 3 letters, 4 digits)
generic letter, generic number, L-junctions of sides
characters, plate sides
parts of characters, parts of plate sides

Figure 2.2: Semantic hierarchy for plate-reading application.

okay and each digit does have its right pieces.

Most of the proposed generative models in the literature share this Markov property ([11], [17], [23], [27], [30], [31]), which makes them tractable. A straight analogy with the Markov backbone under the image setting is the PCFG (Probabilistic Context Free Grammar) model under the language setting. For example, a sample sentence generated from PCFG doesn't make much sense semantically, but it does have a proper syntactic structure. They are all equipped with some versions of dynamic programming machinery. In short, the family of Markov models has its computational advantage at the expense of a common critical shortcoming, that is, too broad coverage. To be more specific, under the Markov backbone (see right panel of Figure 2.1), brick β can be mapped down into a region in the image space by recursively mapping the collection of all its children bricks. Brick β can be activated without caring about its detailed instantiation, as long as certain context-free rules among its parts are met in this region. Such context-free rules guarantee that you have the right pieces; lack of the detailed information of composition (e.g information on relative positions among parts) guarantees that you generate a "disfigured" 4-digit sample.

Such broad-coverage property works against selectivity (also known as invariance versus selectivity dilemma) and ROC performance in any recognition task. It seems one quick way to fix this is to expand the state space such that the state of each brick contains detailed information about its instantiation. For example, the state of a 4-digit brick encodes the fine positional information of all its terminal parts. Theoretically it is feasible, but practically it is unmanageable after only a few layers, even if we only consider the positional attribute. Ideally, we may have other attributes, like scale, angle, color, etc.

2.1.2 Non-Markov Perturbation

Under the Markov backbone, we have the invariance built in without cost. To incorporate the selectivity, we introduce a sequence of attributes through a sequence of non-Markovian perturbations on the current model, starting with the Markov backbone. The attribute associated with a brick β is formally defined as an attribute function of the state configuration of all bricks, $a^{\beta}(I)$, which measures how good the fit is among children parts that instantiate β in this particular interpretation I. For example, consider a 4-digit brick β, $a^{\beta}(I)$ returns the relative distances among the four digits which instantiate brick β in I; similarly consider a character "0" brick α, $a^{\alpha}(I)$ computes the relative distances among the three terminal parts (see Figure 2.3) which are composed into brick α in I. A pair of distributions over attributes associated with each brick β (in the form of likelihood ratio) are needed to form a compositional system, besides the Markov backbone. The first (composed) distribution (which appears as the numerator of the likelihood ratio in Eqn 2.2) p^c_{β} describes the true pattern of an attribute of an object, given that object (represented by β) is present under the perturbed distribution; the second (null) distribution p^0_{β} (which appears as the denominator of the likelihood ratio in Eqn 2.2) describes the same attribute of an object, given that object is present under the current distribution (i.e before the perturbation). Assume the composed distribution has some kind of bellshape with thin-tail property and the null distribution has peak-shape with fat-tail property, their ratio gives us a hint of whether the presence of the children bricks (parts) of β is due to chance or not in the particular interpretation I. The bigger the ratio, the more probable the presence of the parts are due to the presence of the object (which is composed of those parts) and not due to the chance.

In a compositional system,

$$P(I) \propto \frac{\prod_{\beta \in \mathbf{B}} (\epsilon^{\beta}_{x^{\beta}})}{\prod_{\beta \in \mathbf{B}(I)} (1 - \epsilon^{\beta}_{0})} \prod_{\beta \in \mathbf{A}(I)} \frac{p^{c}_{\beta}(a^{\beta}(I))}{p^{0}_{\beta}(a^{\beta}(I))} \tag{2.2}$$

where $\mathbf{A}(I)$, the "above set," is the set of non-terminal *on* (active) bricks in I. The proportionality sign ($\propto$) can be replaced with equality ($=$) if, at the introduction of each attribute function, a^{β}, care is taken to ensure that $p^{0}_{\beta}(a^{\beta})$ is exactly the current ("unperturbed") conditional distribution on a^{β} given $x^{\beta} > 0$. In general, it is not practical to compute an exact null distribution and P must be re-normalized.

The detailed derivation of Eqn 2.2 can be found in [16]. A quick understanding of this model can be achieved through a sampling scheme. To generate a sample under this distribution, first we generate n samples from the Markov backbone, then weigh each sample with a product of likelihood ratios of its attributes. Re-sampling according to the normalized weights as n goes to infinity, we get a context-sensitive sample, i.e., a sample under the compositional distribution. The non-Markov perturbation can be treated as a reweighting of the initial Markov structure, based on the fine compositional information.

To see what that perturbation yields, a compositional 4-digit sample generated through a variant of important sampling is shown in the right panel of Figure 2.3 (for sampling details, see §4.4.1). As we can see, dramatic improvement is achieved under the non-Markovian model, but with an associated cost. In §4.2, we will show that the dependency graph structure under the non-Markovian setting is further complicated, and for a toy example, searching for the optimal interpretation is provable NP-hard

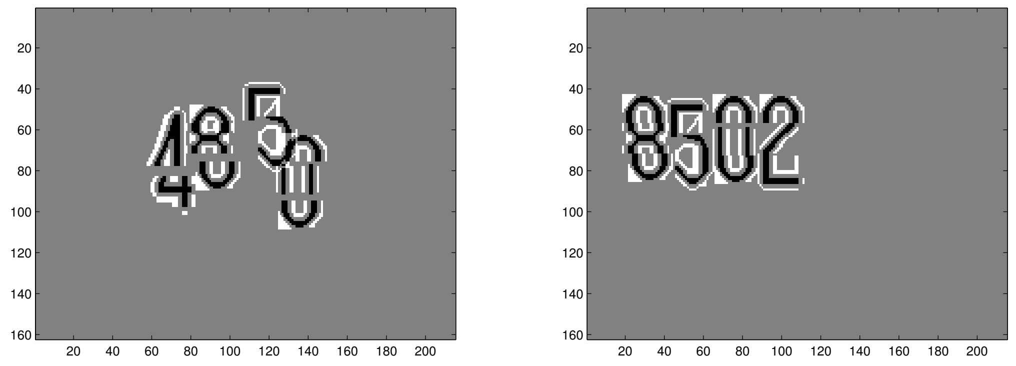


Figure 2.3: Samples from Markov backbone (left panel, '4850') and compositional distribution (right panel, '8502').

(in §2.4). Although the dynamic programming machinery is no longer with us, certain coarse-to-fine computational engine that fits the position quite well is still available.

2.2 Data Model

We briefly describe the data model to complete the Bayesian framework. In a compositional machine, we generally assume the data distribution, conditioned on the interpretation, is a function of the states of the terminal bricks, i.e.,

$$P(\vec{y}|I) = P(\vec{y}|\{x^{\beta} : \beta \in \mathcal{T}\}) \tag{2.3}$$

where $\vec{y}$ is gray-level image and $\mathcal{T} \subseteq \mathbf{B}$ is the set of terminal bricks.

In the license demonstration system, terminal brick refers to reusable parts of characters which could be seen from the left panel of Figure 2.3. The digits "8" and "0" each consist of three parts and digits "4" and "5" each consist of two parts. Instead of running a correlation-based data analysis, for each black on white part filter, we compute its rank sum statistic R, which is defined as the rank sum of the intensities of the black pixels among the union of intensities of black and white pixels, across the image to achieve certain photometric invariance. We assume distribution of intensities of pixels (referenced by active terminal bricks) depends only on R (i.e., R is sufficient) and model R with an exponential distribution. Pixels that are not referenced by any active terminal brick are assumed to be uniformly and independently distributed. Smaller R implies better fit. For more details, please see §4.3.1.

2.3 Parsing Scene

Under the Bayesian framework, the posterior distribution over the interpretations given image, represented by $\vec{y}$, is

$$P(I|\vec{y}) = \frac{P(\vec{y}|I)P(I)}{P(\vec{y})} \propto P(\vec{y}|I)P(I) \tag{2.4}$$

The interpretation I provides a rich semantic and syntactic analysis (parsing) of the scene using the brick variables (vocabulary) defined in the compositional machine, from which each pixel can be identified as simple background (with no structure), or cluttered background (with structure), or objects at different complexity levels. For example, in the license demonstration system, from interpretation I we can read out the detailed locations and identifications of license plate, string, boundary, partial string, L-junction, character, line, and parts, etc., which may occur anywhere across the image.

Equipped with the prior model and data model, ideally we should compute the MAP estimator exactly. But given the sample space we are dealing with (recall, a compositional machine often contains millions of bricks), exact inference is intractable and we have to resort to some computationally feasible approximation.

The computational engine we want to explore is the coarse-to-fine search strategy. It is biologically motivated and has compelling theoretical and empirical evidence in similar framework [7]. We will focus on describing a bottom-up pass (through a compositional machine), aiming at scene interpretation, and touch on a general depth-first search strategy, aiming at fast object detection.

In the bottom-up pass, simple likelihood ratio tests (of presence versus null) for terminal parts are first conducted across the whole image to collect evidence for state configuration of all the terminal bricks. The state of each active terminal brick signals its presence at a particular location. A conservative test threshold is set to ensure no-missed parts of target, but at the cost of many false positives parts. After storing all the candidate parts into an index list, we move to higher layers sequentially. Based on the information (states of bricks) we have from the lower layers, we conduct similar likelihood ratio tests (of composition versus non-composition) incorporating context information to decide whether to compose relevent parts (at lower layer) into objects (at current layer) or not. This top-down contexture information helps to explain away the low-layer ambiguity and propagate the true positives. Again, generated candidate objects are put into the index list before moving upward. This process ends when no further composition is possible. After we are done with one bottom-up pass, the index list consists of a huge number of parts and objects at different complexity levels, each corresponding to a consistent subgraph and equipped with a fitness measure. In a sense, the index list is an over-complete representation of the scene at many levels using the variables in the vocabulary. We next employ a greedy search algorithm to pick candidate objects from the list sequentially to cover the scene. The procedure continues until no object/part left in the list can further improve the interpretation.

The bottom-up pass is referred to as list generation (in §4.3.3.1) and indexing the scene using the elements in the list is referred to as list optimization (in §4.3.3.2). An example of indexed scene is given in Figure 2.4. As you can see, the scene is annotated by only string-related variables (note: the complete state configuration of all bricks can't be inferred from the picture). By accessing the internal representation of the parse, we find that it contains only one string object "307 XXL," very few false postive partial strings and quite a few false positive characters.

Although the above procedure does explore $\mathcal{I}$ in a coarse-to-fine fashion, it is still slow due to the fact that the overwhelming majority of running time is spent on places

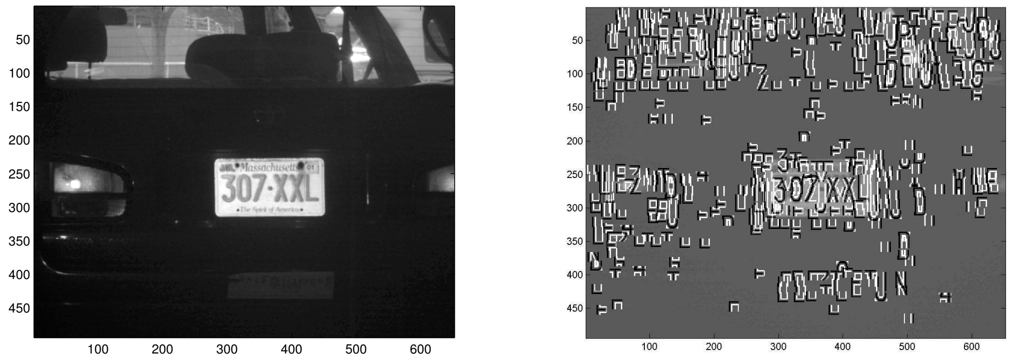


Figure 2.4: Original scene and its parsed scene with string-related variables

that are unnecessary for target detection purpose. With the goal of object detection in mind, we propose a depth-first search scheme over the same compositional architecture, which efficiently allocates computational resource to the most "interesting" regions at the least cost. The idea is to identify an ordering of the same tests as above under which we can make an early decision on the presence of the target. For example, early detection of a license boundary can signal potential license reading in a smaller region and excludes the possibility of presence of a license object in a wide, no-boundary region. The tremendous savings in computation is achieved without sacrificing performance. A self-explaining example is shown in Figure 2.5. For more details, see §4.4.5.

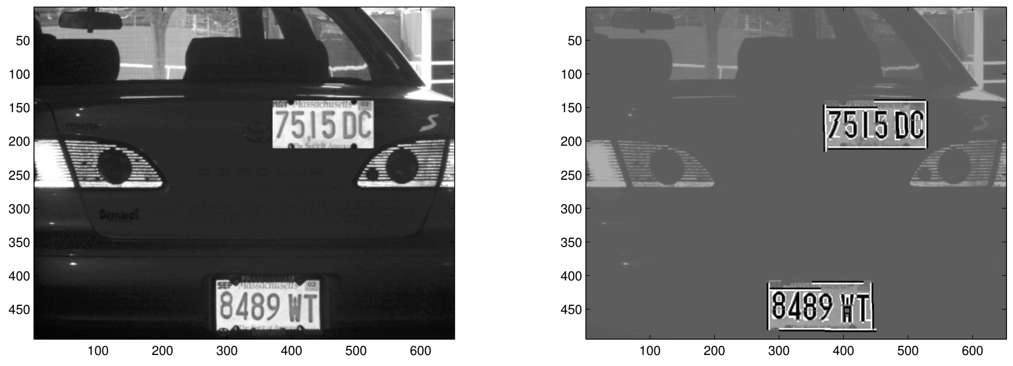


Figure 2.5: Original scene and its parsed scene with license objects only

2.4 Proof of NP-hardness

Under the compositional system, given a test image and a noise model between the image and the terminal (first-layer) bricks, it is computationally intractable to find the most probable state configuration of the underlying hierarchical structure (i.e. the most probable parse) for that image. Precisely, it is a NP hard problem. We will show that there is a correspondence between our problem and the classical set covering problem by a construction argument. Once you can solve our problem, you can solve the set covering problem in polynomial time. We know set covering problem is NP complete, so our problem is NP hard.

Construction Argument

Lemma: A special case of our problem is equivalent to the set covering problem.

Proof: We begin by stating the set covering problem, then from that we construct a toy version of our problem with a two-layer hierarchical structure, and finally show that solving the toy problem for a fixed set of model parameters leads to solving the set covering problem.

Consider a set covering problem; given $S = \{S_i\}_{i \in T}$ a finite collection of sets, each S_i is associated with a positive cost w_i, find $C \subseteq S$ with minimum cost $\sum_{S_i \in C} w_i$ s.t $\bigcup_{S_i \in C} S_i = \bigcup_{S_j \in S} S_j$. This problem is NP complete, because it is within the family of problems which can't be solved by existing algorithms within polynomial time (in terms of the number of sets in S, i.e $|T|$) in general (or worst) case.

Consider a two-layer hierarchical structure, at the terminal layer (first layer), the number of bricks is equal to $|\bigcup_{S_i \in S} S_i|$ (i.e. the size of the set). Each terminal brick represents an element in $\bigcup_{S_i \in S} S_i$, which takes binary values, standing for the presence or absence of that corresponding element. At layer two, there are $|T|$ bricks. For each brick α at layer two, set its i^{th} children set $C_i^\alpha = S_i$, that is, brick α selects (is connected to) the set of terminal bricks representing the elements in S_i when $X^\alpha = i$.

Assume bricks at layer two are $i.i.d$ taking values in $\{0, 1, ..., |T|\}$ with probability

$$\begin{aligned} P(X^\alpha = i) &= \frac{1}{Z} e^{-w_i} \; i = 1, 2, ...|T| \\ P(X^\alpha = 0) &= \frac{1}{Z} \quad \text{where } Z = \sum_{i=1}^{|T|} e^{-w_i} + 1 \end{aligned}$$

A sample image $\vec{Y}$ can be generated in a top-down fashion, that is, pick the state for each layer-two brick independently according to its state probability vector; next, pick the state for each terminal brick τ, which is not referenced by any layer-two brick, identically and independently by the Bernoulli distribution with $\epsilon_1^\tau \doteq P(X^\tau = 1)$ (those referenced terminal bricks have state 1); finally, by applying i.i.d flip noise with probability δ of flip on each of the terminal bricks, we get a binary image $\vec{Y}$.

Suppose we can solve our problem in a general setting (i.e. find the optimal interpretation for any given $\vec{Y}$), we can solve the toy problem in a special case with the input $\vec{Y}$ being all 1's and flip noise $\delta = 0$. That tells us any interpretation with positive probability will have its terminal bricks all set to be 1. Furthermore, if we set the active probability of each terminal brick τ, ϵ_1^τ, to be small enough, (namely, $\epsilon_1^\tau < e^{-\max_i w_i}$), the optimal configuration of layer-two bricks will guarantee to cover all the terminal bricks by the Markov backbone in Eqn 2.1.

Let $X^* = \{\vec{X} : \vec{X}^1 = \vec{1} \wedge \bigcup_{\alpha \in \mathcal{B}_2} C^\alpha_{x^\alpha} = \bigcup_{S_i \in S} S_i\}$, where $\vec{X}^i$ is the state configuration for layer-i bricks, $\mathcal{B}_2$ is the set of layer-two bricks, $C^\alpha_{x^\alpha}$ is the children set of brick α when it takes value x^α and X^* is the set of all possible state configurations which meet the two constraints. Then,

$$\begin{aligned}
\arg\max_{\vec{X}} P(\vec{X}|\vec{Y}=\vec{1}) &= \arg\max_{\vec{X}\in X^*} P(\vec{X}|\vec{Y}=\vec{1}) \\
&\overset{(1)}{=} \arg\max_{\vec{X}\in X^*} P(\vec{X}|\vec{X}^1=\vec{1}) \\
&= \arg\max_{\vec{X}\in X^*} P(\vec{X}) \\
&\overset{(2)}{=} \arg\max_{\vec{X}\in X^*} \frac{P(\vec{X}^2)P(\vec{X}^1=\vec{1})}{P(\vec{X}^1=\vec{1})} \\
&= \arg\max_{\vec{X}\in X^*} P(\vec{X}^2) \\
&\overset{(3)}{=} \arg\max_{\vec{X}\in X^*} (\frac{1}{Z})^{|T|}\, exp(-\sum_{i\in V(\vec{X}^2)} w_i) \\
&= \arg\max_{\vec{X}\in X^*} (-\sum_{i\in V(\vec{X}^2)} w_i) \\
&= \arg\min_{\vec{X}\in X^*} \sum_{i\in V(\vec{X}^2)} w_i \\
&= \arg\min_{C\subseteq S} \sum_{S_i\in C} w_i \quad \text{where } \bigcup_{S_i\in C} S_i = \bigcup_{S_j\in S} S_j
\end{aligned}$$

Equality (1) holds because $\vec{X}^1 = \vec{Y}$ almost surely. Equality (2) is from the definition of the context-free probability (see eqn 2.1). In equality (3), $V(\vec{X}^2)$ is the set of active states occured in $\vec{X}^2$. The last equality abused the notation a little bit, since the arguments to achieve the same minimun value are of different types. It basically says if we know the optimal configuration of layer-two bricks that covers all active layer-one bricks (i.e solve the toy program), the collection of their corresponding chosen children sets is the optimal solution to the covering problem.

Note:

1. The optimal solution for our problem will not allow complete overlap by setting $P(X^\alpha = 0) > P(X^\alpha = i)$, for $i > 0$. In set covering situation, optimal solution won't contain identical subset S_i either.

2. Given we can solve set covering problem, it does not necessarily imply an optimal solution to our problem. In a general setting, the most probable configuration of layer-two bricks may not cover all terminal bricks.

Chapter 3

Theoretical Evidence for Hierarchical Representation

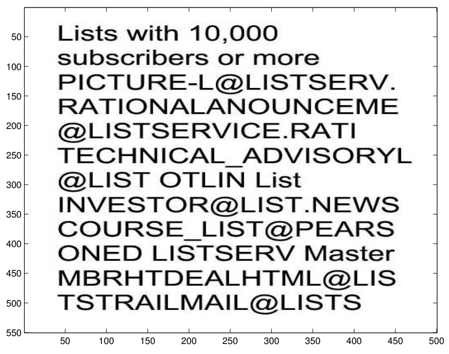
3.1 Introduction

Human vision is both highly invariant (invariance refers to the extent to which objects are detected independently of their presentations) and highly selective (selectivity refers to avoiding the misclassification of other structures). Computer vision systems are rarely both. In particular, systems which are highly invariant often suffer from an unacceptable number of false positives. Such an "invariance and selectivity" dilemma is due to the extreme variations in objects and background presentations. In this chapter, we demonstrate the efficient discrimination nested in the hierarchical representation through a theorem justifying a better ROC curve. We first present a thought experiment, then formalize it and show some theoretical results.

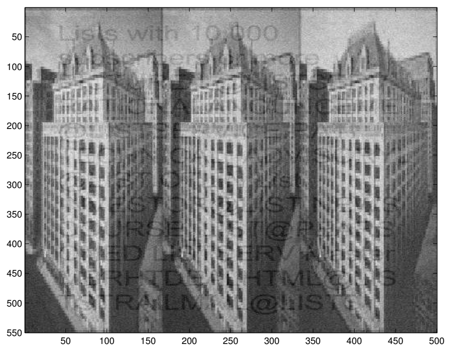
3.2 Thought Experiment

We know any vision task can be treated as a hypothesis testing problem. Consider such a thought experiment. The goal is to detect all the "L's" in a background full of noise (both structured and non-structured) as seen in Figure 3.1. Hypothesis H_0, H_1, H_2, and H_3 are tested for a null component, a vertical part, a horizontal part and a whole L, all with an additive noise in the test image. The generative model involved is $Y = .1X_1 + .6X_2 + .3Z$, where X_1 is the original clean text file with 19 "L's" inside it, X_2 is a building image (structured noise), Z is Gaussian noise with a mean of 0, standard deviation of 30, and Y is the output image.

The optimal test from the Neyman-Pearson lemma declares "L" if $\{\frac{L(Y|H_3)}{L(Y|\bar{H}_3)} > c\}$ (i.e., the data likelihood ratio under H_3 versus $\bar{H}_3$ passes some threshold). It is not practical to enumerate all the $\bar{H}_3$ in general, instead we test against a universal null, i.e., declare "L" if $\{\frac{L(Y|H_3)}{L(Y|H_0)} > c\}$ (also called testing under the whole model). Since clutter background shares the same reusable parts as the target, the idea of compositionality suggests sequential tests to explain away all the ambiguity caused by treating everything as "L" or non "L." So the suggested test declares a "L" if



X_1



$\mathbf{Y}$

Figure 3.1: The original clean text and its noisy version

$\{\frac{L(Y|H_1)}{L(Y|H_0)} > c \quad and \quad \frac{L(Y|H_2)}{L(Y|H_0)} > c\}$ (also called testing under the parts model).

The likelihood ratio test is based on rank-sum statistics instead of raw data, but they are equivalent under the assumption of sufficiency (see Geman [18]). For further information about rank-sum statistics, see §4.3.1.1. The ROC curves for the test image are as follows.

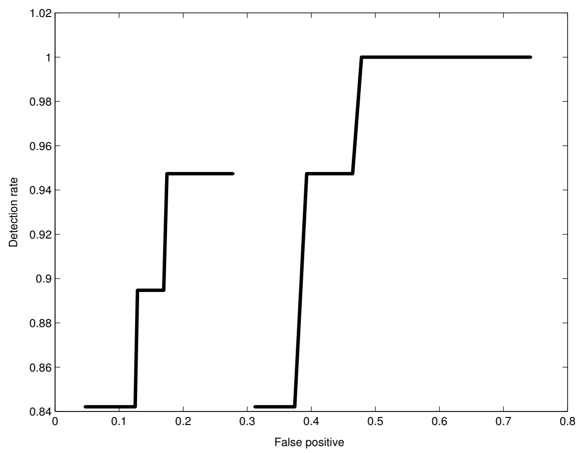


Figure 3.2: The left curve, based on the parts model, the right curve on the whole model for the thought experiment

Note: The curves are not smooth, because they are computed from a fixed image with a discrete number of "L" in it.

3.3 Theoretical Results

3.3.1 Normal Noise

Assume the target is composed of two parts. The related generative model is $Y = X + \sigma * Z$, where X takes values in (0,0),(1,0),(0,1),(1,1) (which stand for nothing, part 1 present, part 2 present, the whole target present) with probability p_0, p_1, p_2, p_3, respectively $(p_0 + p_1 + p_2 + p_3 = 1)$, and Z is some continuous random variable with the support of its density function over all R and meet certain tail property.

To get a feeling of what the limiting ROC (as $\sigma \to 0$) looks like in either case (i.e., under whole and parts model), let's first consider the normal case, that is, Z is a vector of two i.i.d standard normal random variables. Some related notations are listed below.

$$\begin{aligned}
T_0(Y,c) &\doteq 1_{\{\frac{L(Y|X=(1,1))}{L(Y|X=(0,0))}>c\}} \\
T_p(Y,c) &\doteq 1_{\{\frac{L(Y|X_1=1)}{L(Y|X_1=0)}>c \quad \& \quad \frac{L(Y|X_2=1)}{L(Y|X_2=0)}>c\}}
\end{aligned}$$

$$\begin{aligned}
R_0(c) &\doteq P(T_0(Y,c)=1|X=(1,1)) \\
S_0(c) &\doteq P(T_0(Y,c)=1|X\neq(1,1)) \\
R_p(c) &\doteq P(T_p(Y,c)=1|X=(1,1)) \\
S_p(c) &\doteq P(T_p(Y,c)=1|X\neq(1,1))
\end{aligned}$$

Under the normal case, we can compute these terms explictly.

$$\begin{aligned}
R_0(c) &= P(\frac{P(Y|X=(1,1))}{P(Y|X=(0,0))} > c|X=(1,1)) \\
&= P(\log(\frac{P(Y|X=(1,1))}{P(Y|X=(0,0))}) > \log c|X=(1,1)) \\
&= P(\frac{Y_1^2+Y_2^2-(Y_1-1)^2-(Y_2-1)^2}{2\sigma^2} > \log c|X=(1,1)) \\
&= P(Y_1+Y_2 > 1+\sigma^2\log c|X=(1,1)) \\
&= P(\frac{Y_1-1}{\sigma}+\frac{Y_2-1}{\sigma} > \sigma\log c-\frac{1}{\sigma}|X=(1,1)) \\
&= P(Z_1+Z_2 > \sigma\log c-\frac{1}{\sigma})
\end{aligned}$$

$$\begin{aligned}
S_0(c) &= P(\frac{P(Y|X=(1,1))}{P(Y|X=(0,0))} > c|X\neq(1,1)) \\
&= P(Y_1+Y_2 > 1+\sigma^2\log c|X\neq(1,1)) \\
&= \frac{p_0}{p_0+p_1+p_2}P(Y_1+Y_2 > 1+\sigma^2\log c|X=(0,0)) \\
&\quad +\frac{p_1}{p_0+p_1+p_2}P(Y_1+Y_2 > 1+\sigma^2\log c|X=(1,0)) \\
&\quad +\frac{p_2}{p_0+p_1+p_2}P(Y_1+Y_2 > 1+\sigma^2\log c|X=(0,1)) \\
&= \frac{p_0}{p_0+p_1+p_2}P(Z_1+Z_2 > \frac{1}{\sigma}+\sigma\log c)+\frac{p_1+p_2}{p_0+p_1+p_2}P(Z_1+Z_2 > \sigma\log c)
\end{aligned}$$

Similiarly,

$$\begin{aligned}
R_p(c) &= P(Z_1 > -\frac{1}{2\sigma}+\sigma\log c)P(Z_2 > -\frac{1}{2\sigma}+\sigma\log c) \\
&= [P(Z_1 > -\frac{1}{2\sigma}+\sigma\log c)]^2
\end{aligned}$$

$$\begin{aligned}
S_p(c) &= \frac{p_0}{p_0+p_1+p_2}[P(Z_1 > \frac{1}{2\sigma}+\sigma\log c)]^2 \\
&\quad +\frac{p_1+p_2}{p_0+p_1+p_2}P(Z_1 > \frac{1}{2\sigma}+\sigma\log c)P(Z_2 > -\frac{1}{2\sigma}+\sigma\log c)
\end{aligned}$$

Fix c, as σ goes to 0, $(R_0, S_0) \to (1, \frac{p_1+p_2}{2(p_0+p_1+p_2)})$ and $(R_p, S_p) \to (1, 0)$. We know both ROCs go through the points (0,0) and (1,1). It seems that the limiting ROC for the whole model is different from the parts model which goes along the y-axis (i.e., R-axis) and then jumps to (1,1). We conjecture that there is a nontrivial region up in the corner above the family of all ROC curves generated by any $\sigma > 0$ and general noise Z as above under the whole model. It turns out this is not true. Here is a proof by contradiction for the normal case.

Proof: Suppose the conjecture is true; there always exist a 45 degree line $R - S = k$ which cross only that non-trivial region and $k < 1$, i.e., $R_0(c) - S_0(c) \leq k < 1$ for any $\sigma > 0$ and any $c > 0$.

In this normal case, let $a = \frac{1}{\sigma}, b = \sigma \log c$; $R_0(c) - S_0(c) = P(Z_1 + Z_2 > b - a) - \frac{p_0}{p_0+p_1+p_2} P(Z_1 + Z_2 > a + b) - \frac{p_1+p_2}{p_0+p_1+p_2} P(Z_1 + Z_2 > b)$. Choose $c = e^{\frac{1}{\epsilon^3}}, \sigma = \epsilon^2$, we have $a = \frac{1}{\epsilon^2}, b = \frac{1}{\epsilon}, b - a = \frac{1}{\epsilon} - \frac{1}{\epsilon^2}$. Let $\sigma \to 0$, then we have $b \to \infty, a \to \infty, b - a \to -\infty$. This implies $R_0(c) - S_0(c) \to 1$, meaning there is no such $k < 1$ to define the line. Contradiction. $\square$

This tells us that as the noise goes to 0, we can choose the thresholds to diverge in a certain way such that the generated ROC curve under the whole model can pass through any point near the upper-left corner just as under the parts model.

This fact suggests that we have to consider relative things, like ratio, instead of absolute things. Here is one formulation.

Theorem 3.3.1: *Under the normal case, for fixed* $1 > r > 0$, $\forall M > 0$, $\exists \epsilon > 0$ *s.t* $\forall \sigma < \epsilon$, $\frac{S_0(R_0^{-1}(r))}{S_p(R_p^{-1}(r))} > M$.

Proof: let $Z^0 \doteq Z_1 + Z_2$, and z_r^0, z_r^1 are the numbers such that $P(Z^0 > z_r^0) = r$, $P(Z_1 > z_r^1) = r$ respectively. Bearing in mind the formula we have derived for R_0, S_0, R_p, S_p, we have

$$r = R_0(c_0) = P(Z_1 + Z_2 > -\frac{1}{\sigma} + \sigma \log c_0) \Rightarrow -\frac{1}{\sigma} + \sigma \log c_0 = z_r^0$$

$$r = R_p(c_p) = [P(Z_1 > -\frac{1}{2\sigma} + \sigma \log c_p)]^2 \Rightarrow -\frac{1}{2\sigma} + \sigma \log c_p = z_{\sqrt{r}}^1$$

$$\begin{aligned} S_0(c_0) &= \frac{p_0}{p_0+p_1+p_2} P(Z_1 + Z_2 > \frac{2}{\sigma} + z_r^0) \\ &\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(Z_1 + Z_2 > \frac{1}{\sigma} + z_r^0) \end{aligned}$$

$$\begin{aligned} S_p(c_p) &= \frac{p_0}{p_0+p_1+p_2} [P(Z_1 > \frac{1}{\sigma} + z_{\sqrt{r}}^1)]^2 \\ &\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(Z_1 > \frac{1}{\sigma} + z_{\sqrt{r}}^1) P(Z_1 > z_{\sqrt{r}}^1) \end{aligned}$$

$$\begin{aligned} \frac{S_0(R_0^{-1}(r))}{S_p(R_p^{-1}(r))} &\geq \frac{\frac{p_1+p_2}{p_0+p_1+p_2} P(Z_1 + Z_2 > \frac{1}{\sigma} + z_r^0)}{(\frac{p_0}{p_0+p_1+p_2} + \frac{p_1+p_2}{p_0+p_1+p_2} \cdot \sqrt{r}) P(Z_1 > \frac{1}{\sigma} + z_{\sqrt{r}}^1)} \\ &= C \cdot \frac{P(\frac{Z^0}{\sqrt{2}} > \frac{1}{\sqrt{2}} \cdot (\frac{1}{\sigma} + z_r^0))}{P(Z_1 > \frac{1}{\sigma} + z_{\sqrt{r}}^1)} \\ &> C \cdot \frac{(\sqrt{2}(\frac{1}{\sigma} + z_r^0)^{-1} - (\sqrt{2})^3(\frac{1}{\sigma} + z_r^0)^{-3}) \frac{1}{\sqrt{2\pi}} \exp(-\frac{(\frac{1}{\sigma}+z_r^0)^2}{2 \cdot 2})}{(\frac{1}{\sigma} + z_{\sqrt{r}}^1)^{-1} \frac{1}{\sqrt{2\pi}} \exp(-\frac{(\frac{1}{\sigma}+z_{\sqrt{r}}^1)^2}{2})} \\ &= C \cdot (\frac{1}{\sigma} + z_{\sqrt{r}}^1) \cdot (\sqrt{2}(\frac{1}{\sigma} + z_r^0)^{-1} - (\sqrt{2})^3(\frac{1}{\sigma} + z_r^0)^{-3}) \\ &\quad \cdot \exp(\frac{2(\frac{1}{\sigma} + z_{\sqrt{r}}^1)^2 - (\frac{1}{\sigma} + z_r^0)^2}{4}) \end{aligned}$$

The second inequlity is guaranteed by $(x^{-1} - x^{-3})f(x) < P(X > x) < x^{-1}f(x)$, when $X \sim N(0,1)$, $f(x)$ is normal density and $x > 0$ (so we need small σ to ensure $x > 0$ for arbitrary r).

Since $(\frac{1}{\sigma} + z_{\sqrt{r}}^1)(\sqrt{2}(\frac{1}{\sigma} + z_r^0)^{-1} - (\sqrt{2})^3(\frac{1}{\sigma} + z_r^0)^{-3}) \to \sqrt{2}$ as $\sigma \to 0$ and since $\exp(\frac{2(\frac{1}{\sigma}+z_{\sqrt{r}}^1)^2 - (\frac{1}{\sigma}+z_r^0)^2}{4}) = \exp(\frac{\frac{1}{\sigma^2} + \frac{1}{\sigma} \cdot d_1 + d_2}{4}) \to \infty$ as $\sigma \to 0$, we complete the proof. $\Box$

3.3.2 Generalized Noise

To generalize the result, we first derive $R_0(c), R_p(c), S_0(c), S_p(c)$ in the general setting. Given independent noise $Z = (Z_1, Z_2)$, when $X = (1,1), Y = (Y_1, Y_2) = (1,1) + \sigma \cdot (Z_1, Z_2)$, by defnition. It's easy to derive $f_Y(y_1, y_2) = \frac{1}{\sigma^2} f_{Z_1}(\frac{y_1-1}{\sigma}) f_{Z_2}(\frac{y_2-1}{\sigma})$ when $X = (1,1)$. Similiarly, we have

$$\begin{aligned}
f_Y(y_1,y_2) &= \frac{1}{\sigma^2} f_{Z_1}(\frac{y_1}{\sigma}) f_{Z_2}(\frac{y_2}{\sigma}) \quad when \quad X = (0,0) \\
f_Y(y_1,y_2) &= \frac{1}{\sigma^2} f_{Z_1}(\frac{y_1-1}{\sigma}) f_{Z_2}(\frac{y_2}{\sigma}) \quad when \quad X = (1,0) \\
f_Y(y_1,y_2) &= \frac{1}{\sigma^2} f_{Z_1}(\frac{y_1}{\sigma}) f_{Z_2}(\frac{y_2-1}{\sigma}) \quad when \quad X = (0,1)
\end{aligned}$$

$$\begin{aligned}
R_0(c_0) &= P(\frac{P(Y|X=(1,1))}{P(Y|X=(0,0))} \geq c_0 | X = (1,1)) \\
&= P_Y(\frac{f_{Z_1}(\frac{Y_1-1}{\sigma}) f_{Z_2}(\frac{Y_2-1}{\sigma})}{f_{Z_1}(\frac{Y_1}{\sigma}) f_{Z_2}(\frac{Y_2}{\sigma})} \geq c_0 | X = (1,1)) \\
&= P_Z(\frac{f_{Z_1}(Z_1) f_{Z_2}(Z_2)}{f_{Z_1}(Z_1 + \frac{1}{\sigma}) f_{Z_2}(Z_2 + \frac{1}{\sigma})} \geq c_0)
\end{aligned}$$

$$\begin{aligned}
S_0(c_0) &= P_Y(\frac{P(Y|X=(1,1))}{P(Y|X=(0,0))} \geq c_0 | X \neq (1,1)) \\
&= \frac{p_0}{p_0+p_1+p_2} P(\frac{f_{Z_1}(\frac{Y_1-1}{\sigma}) f_{Z_2}(\frac{Y_2-1}{\sigma})}{f_{Z_1}(\frac{Y_1}{\sigma}) f_{Z_2}(\frac{Y_2}{\sigma})} \geq c_0 | X = (0,0)) \\
&\quad + \frac{p_1}{p_0+p_1+p_2} P_Y(\frac{f_{Z_1}(\frac{Y_1-1}{\sigma}) f_{Z_2}(\frac{Y_2-1}{\sigma})}{f_{Z_1}(\frac{Y_1}{\sigma}) f_{Z_2}(\frac{Y_2}{\sigma})} \geq c_0 | X = (1,0)) \\
&\quad + \frac{p_2}{p_0+p_1+p_2} P_Y(\frac{f_{Z_1}(\frac{Y_1-1}{\sigma}) f_{Z_2}(\frac{Y_2-1}{\sigma})}{f_{Z_1}(\frac{Y_1}{\sigma}) f_{Z_2}(\frac{Y_2}{\sigma})} \geq c_0 | X = (0,1)) \\
&= \frac{p_0}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1 - \frac{1}{\sigma}) f_{Z_2}(Z_2 - \frac{1}{\sigma})}{f_{Z_1}(Z_1) f_{Z_2}(Z_2)} \geq c_0) \\
&\quad + \frac{p_1}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1) f_{Z_2}(Z_2 - \frac{1}{\sigma})}{f_{Z_1}(Z_1 + \frac{1}{\sigma}) f_{Z_2}(Z_2)} \geq c_0) \\
&\quad + \frac{p_2}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1 - \frac{1}{\sigma}) f_{Z_2}(Z_2)}{f_{Z_1}(Z_1) f_{Z_2}(Z_2 + \frac{1}{\sigma})} \geq c_0)
\end{aligned}$$

Similiarly, we have

$$\begin{aligned}
R_p(c_1,c_2) &= P(\frac{P(Y|X_1=1)}{P(Y|X_1=0)} \geq c_1 \quad \& \quad \frac{P(Y|X_2=1)}{P(Y|X_2=0)} \geq c_2 | X=(1,1)) \\
&= P_Y(\frac{f_{Z_1}(\frac{Y_1-1}{\sigma})}{f_{Z_1}(\frac{Y_1}{\sigma})} \geq c_1 | X_1 = 1) P_Y(\frac{f_{Z_2}(\frac{Y_2-1}{\sigma})}{f_{Z_2}(\frac{Y_2}{\sigma})} \geq c_2 | X_2 = 1) \\
&= P_Z(\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1+\frac{1}{\sigma})} \geq c_1) P_Z(\frac{f_{Z_2}(Z_2)}{f_{Z_2}(Z_2+\frac{1}{\sigma})} \geq c_2)
\end{aligned}$$

$$\begin{aligned}
S_p(c_1,c_2) &= P(\frac{P(Y|X_1=1)}{P(Y|X_1=0)} \geq c_1 \quad \& \quad \frac{P(Y|X_2=1)}{P(Y|X_2=0)} \geq c_2 | X \neq (1,1)) \\
&= \frac{p_0}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1-\frac{1}{\sigma})}{f_{Z_1}(Z_1)} \geq c_1) P_Z(\frac{f_{Z_2}(Z_2-\frac{1}{\sigma})}{f_{Z_2}(Z_2)} \geq c_2) \\
&\quad + \frac{p_1}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1+\frac{1}{\sigma})} \geq c_1) P_Z(\frac{f_{Z_2}(Z_2-\frac{1}{\sigma})}{f_{Z_2}(Z_2)} \geq c_2) \\
&\quad + \frac{p_2}{p_0+p_1+p_2} P_Z(\frac{f_{Z_1}(Z_1-\frac{1}{\sigma})}{f_{Z_1}(Z_1)} \geq c_1) P_Z(\frac{f_{Z_2}(Z_2)}{f_{Z_2}(Z_2+\frac{1}{\sigma})} \geq c_2)
\end{aligned}$$

Theorem 3.3.2: *Given independent noises Z_1, Z_2 with positive, continuous density functions $f_{Z_1}(z)$, $f_{Z_2}(z)$ respectively, if there exists $a > 0, b > 0, \tilde{a} > 0, \tilde{b} > 0, d > 1, d \geq d' > 0$ such that $\lim_{z\to\infty} \frac{-\log(f_{Z_1}(z))}{|z|^d} = a, \lim_{z\to-\infty} \frac{-\log(f_{Z_1}(z))}{|z|^{d'}} = b$ and $\lim_{z\to\infty} \frac{-\log(f_{Z_2}(z))}{|z|^d} = \tilde{a}, \lim_{z\to-\infty} \frac{-\log(f_{Z_2}(z))}{|z|^{d'}} = \tilde{b}$, then $\lim_{\sigma\to 0} \frac{S_0(c_0)}{S_p(c_1,c_2)} = \infty$, while $R_0(c_0) = R_p(c_1,c_2) = r$, for any fixed $0 < r < 1$ and for certain $c_1(r,\sigma), c_2(r,\sigma)$.*

Proof: The layout of the proof is straightforward; we first solve for c_0, c_1, c_2 such that $R_0(c_0) = R_p(c_1,c_2) = r$, then plug the c_0, c_1, c_2 into $S_0(c_0), S_p(c_1,c_2)$ to analyze the ratio as σ goes to 0.

We start by showing the theorem holds when $d = d'$ and Z_1, Z_2 are *i.i.d*, (i.e., $c_1 = c_2$ by symmetry), then we will generalize it at the end of proof.

Let

$$
\begin{aligned}
r_1 &\doteq log\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1+\frac{1}{\sigma})}\\
r_2 &\doteq log\frac{f_{Z_2}(Z_2)}{f_{Z_2}(Z_2+\frac{1}{\sigma})}\\
r_3 &\doteq log\frac{f_{Z_2}(Z_2)}{f_{Z_2}(Z_2-\frac{1}{\sigma})}\\
\mu_1 &\doteq E(r_1) \quad S_1 \doteq s.d(r_1)\\
\mu_2 &\doteq E(r_2) \quad S_2 \doteq s.d(r_2)\\
\mu_3 &\doteq E(r_3) \quad S_3 \doteq s.d(r_3)\\
\phi(z) &\doteq \frac{-\log(f(z))}{|z|^d}
\end{aligned}
$$

Here $s.d$ stands for standard deviation. By assumption, we know $\phi(z)$ is continuous and $\lim_{z\to\infty}\phi(z)=a$, $\lim_{z\to-\infty}\phi(z)=b$. And $f(z)=e^{-\phi(z)|z|^d}$ for $z\neq 0$.

Lemma: $\lim_{\sigma\to 0}\frac{\mu_1}{a\sigma^{-d}}=1$, $\lim_{\sigma\to 0}\frac{S_1}{\sigma^{-d}}=0$. *(Proof in Appendix)*

For any fixed $0<r<1$,

$$
\begin{aligned}
R_0(c_0)=r &\Rightarrow P\Big(\frac{f_{Z_1}(Z_1)f_{Z_2}(Z_2)}{f_{Z_1}(Z_1+\frac{1}{\sigma})f_{Z_2}(Z_2+\frac{1}{\sigma})}\geq c_0\Big)=r\\
&\Rightarrow P(r_1+r_2\geq\log(c_0))=r\\
&\Rightarrow P\Big(\frac{1}{\sqrt{2}}\cdot\Big(\frac{r_1-\mu_1}{S_1}+\frac{r_2-\mu_2}{S_2}\Big)\geq\frac{\log(c_0)-2\mu_1}{\sqrt{2}\cdot S_1}\Big)=r\\
&\Rightarrow \frac{\log(c_0)-2\mu_1}{\sqrt{2}\cdot S_1}=\alpha_r\\
&\Rightarrow \log(c_0)=\sqrt{2}\alpha_r S_1+2\mu_1
\end{aligned}
$$

$$\begin{aligned}
R_p(c_1) = r \quad &\Rightarrow \quad [P(\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1 + \frac{1}{\sigma})} \geq c_1)]^2 = r \\
&\Rightarrow \quad [P(r_1 \geq \log(c_1))]^2 = r \\
&\Rightarrow \quad P(\frac{r_1 - \mu_1}{S_1} \geq \frac{\log(c_1) - \mu_1}{S_1}) = \sqrt{r} \\
&\Rightarrow \quad \frac{\log(c_1) - \mu_1}{S_1} = \beta_{\sqrt{r}} \\
&\Rightarrow \quad \log(c_1) = \beta_{\sqrt{r}} S_1 + \mu_1
\end{aligned}$$

where α_r is the r-quantile point for $\frac{1}{\sqrt{2}} \cdot (\frac{r_1-\mu_1}{S_1} + \frac{r_2-\mu_2}{S_2})$ and $\beta_{\sqrt{r}}$ is the $\sqrt{r}$-quantile point for $\frac{r_1-\mu_1}{S_1}$.

Lemma: *Let W_n be a sequence of continuous RVs with mean 0 and variance 1, for any fixed r $(0 < r < 1)$, consider a sequence of η_n such that $P(W_n \geq \eta_n) = r$, then $|\eta_n| \leq max(\frac{1}{\sqrt{r}}, \frac{1}{\sqrt{1-r}})$, for any n. (Proof in Appendix)*

Since $\frac{1}{\sqrt{2}} \cdot (\frac{r_1-\mu_1}{S_1} + \frac{r_2-\mu_2}{S_2})$ and $\frac{r_1-\mu_1}{S_1}$ are RVs with mean 0 and variance 1, α_r and $\beta_{\sqrt{r}}$ are both bounded by the above lemma (i.e., they are all O(1) terms).

$S_0(c_0)$ and $S_p(c_1)$ can be rewritten as

$$\begin{aligned}
S_0(c_0) &= \frac{p_0}{p_0+p_1+p_2} P(\frac{f_{Z_1}(Z_1 - \frac{1}{\sigma}) f_{Z_2}(Z_2 - \frac{1}{\sigma})}{f_{Z_1}(Z_1) f_{Z_2}(Z_2)} \geq c_0) \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(\frac{f_{Z_1}(Z_1) f_{Z_2}(Z_2 - \frac{1}{\sigma})}{f_{Z_1}(Z_1 + \frac{1}{\sigma}) f_{Z_2}(Z_2)} \geq c_0) \\
&= \frac{p_0}{p_0+p_1+p_2} P(-r_{31} - r_{32} \geq \log(c_0)) \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(r_1 - r_3 \geq \log(c_0)) \quad (\textit{where } f_{r_{31}} = f_{r_{32}} = f_{r_3}) \\
&= \frac{p_0}{p_0+p_1+p_2} P(-r_{31} - r_{32} \geq \sqrt{2}\alpha_r S_1 + 2\mu_1) \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(r_1 - r_3 \geq \sqrt{2}\alpha_r S_1 + 2\mu_1)
\end{aligned}$$

$$
\begin{aligned}
S_p(c_1) &= \frac{p_0}{p_0+p_1+p_2}[P(\frac{f_{Z_1}(Z_1-\frac{1}{\sigma})}{f_{Z_1}(Z_1)} \geq c_1)]^2 \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1+\frac{1}{\sigma})} \geq c_1) P(\frac{f_{Z_2}(Z_2-\frac{1}{\sigma})}{f_{Z_2}(Z_2)} \geq c_1) \\
&= \frac{p_0}{p_0+p_1+p_2} P(-r_3 \geq \log(c_1))^2 \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} P(r_1 \geq \log(c_1)) P(-r_3 \geq \log(c_1)) \\
&= \frac{p_0}{p_0+p_1+p_2} P(-r_3 \geq \beta_{\sqrt{r}} S_1 + \mu_1)^2 \\
&\quad + \frac{p_1+p_2}{p_0+p_1+p_2} \sqrt{r} P(-r_3 \geq \beta_{\sqrt{r}} S_1 + \mu_1)
\end{aligned}
$$

Given the forms of $S_0(c_0), S_p(c_1)$, it is simple to see

$$
\lim_{\sigma \to 0} \frac{S_0(c_0)}{S_p(c_1)} \geq \lim_{\sigma \to 0} C \cdot \frac{P(r_1 - r_3 \geq \sqrt{2}\alpha_r S_1 + 2\mu_1)}{P(-r_3 \geq \beta_{\sqrt{r}} S_1 + \mu_1)}
$$

where $C = \frac{p_1+p_2}{p_0+p_1+p_2} / (\frac{p_0}{p_0+p_1+p_2} + \frac{p_1+p_2}{p_0+p_1+p_2}\sqrt{r})$.

Lemma: $\lim_{\sigma \to 0} \frac{P(r_1 - r_3 \geq 2\mu_1 + \sqrt{2}\alpha_r S_1)}{P(Z_1 \geq \frac{u_\delta}{\sigma}, Z_2 \geq \frac{v_\delta}{\sigma})} \geq 1$, *for some* $u_\delta > 0, v_\delta > 0$ *such that* $u_\delta^d + v_\delta^d < 1$.*(Proof in Appendix)*

Lemma: $\lim_{\sigma \to 0} \frac{P(-r_3 \geq \mu_1 + \beta_{\sqrt{r}} S_1)}{P(Z_2 \geq \frac{1-\eta}{\sigma})} \leq 1$ *for any* $\eta > 0$.*(Proof in Appendix)*

Lemma: $\lim_{k \to \infty} \frac{P(Z \geq k)}{e^{-ak^d + o(k^d)}} = 1$. *(Proof in Appendix)*

Combine the above three lemmas; let $\eta = \frac{1-(u_\delta^d + v_\delta^d)^{\frac{1}{d}}}{2}$ and $\epsilon = \frac{a}{4} \cdot [(\frac{1+(u_\delta^d + v_\delta^d)^{d^{-1}}}{2})^d - u_\delta^d - v_\delta^d]$, we have

$$\begin{aligned}
\lim_{\sigma\to 0}\frac{S_0(c_0)}{S_p(c_1)} &\geq C\cdot\lim_{\sigma\to 0}\frac{P(Z_1\geq\frac{u_\delta}{\sigma}, Z_2\geq\frac{v_\delta}{\sigma})}{P(Z_2\geq\frac{1-\eta}{\sigma})} \\
&= C\cdot\lim_{\sigma\to 0}\frac{P(Z_1\geq\frac{u_\delta}{\sigma})P(Z_2\geq\frac{v_\delta}{\sigma})}{P(Z_2\geq\frac{1-\eta}{\sigma})} \\
&= C\cdot\lim_{\sigma\to 0}\frac{e^{-a(\frac{u_\delta}{\sigma})^d}e^{-a(\frac{v_\delta}{\sigma})^d+o(\frac{1}{\sigma^d})}}{e^{-a(\frac{1-\eta}{\sigma})^d+o(\frac{1}{\sigma^d})}} \\
&\geq C\cdot\lim_{\sigma\to 0}e^{(\frac{1}{\sigma})^d(a(1-\eta)^d-au_\delta^d-av_\delta^d-2\epsilon)} \\
&= \infty
\end{aligned}$$

Lemma: *When the left tail decays at rate $0 < d' \leq d$, all the previous lemmas hold and all the arguments follow.(Proof in Appendix)*

Lemma: *When Z_1, Z_2 are not identical but independent, the theorem holds. (Proof in Appendix)*

With the help of the above two lemmas, the proof for the theorem is complete. $\square$

Lemma: *Consider i.i.d noise with double exponential distribution, i.e., $f(z) = \frac{1}{2}e^{-|z|}$, the ratio is finite. (Proof in Appendix)*

3.4 Extension

Assume a test image is generated from a true image (i.e., a binary image with pixels on a part having equal intensity and pixels off a part also having equal intensity but a different one) with additive independent noise to each pixel. To apply the main theorem to a test image, we have to fill the gap between noise on a part (a set of finite pixels) and noise on its composing individual pixel. It turns out that as long as each additive noise distribution behaves as we described in the theorem (with same decay order, d), the noise distribution on a part has the same form, thus the main theorem can be applied. We justify this by applying the following theorem finite times.

Theorem 3.4.1: *Given* $\lim_{z\to\infty} \frac{-\log f_{Z_1}(z)}{z^d} = C_1 > 0$, $\lim_{z\to\infty} \frac{-\log f_{Z_2}(z)}{z^d} = C_2 > 0$, *then* $\lim_{z\to\infty} \frac{-\log f_Z(z)}{z^d} = C > 0$, *where* $Z = Z_1 + Z_2$ *and* $d > 0$.

Proof: We first derive a lower bound for $\liminf_{z\to\infty} \frac{-\log f_Z(z)}{z^d}$, an upper bound for $\limsup_{z\to\infty} \frac{-\log f_Z(z)}{z^d}$, then show they are equivalent.

WLOG, assume $C_2 \geq C_1$, $P(Z_1 \geq 0)$ and $P(Z_2 \geq 0)$ are stictly positive. $\forall\ \epsilon > 0$, $\forall\ 0 < p < 1, \exists M(\epsilon, p) > 0, s.t\ \frac{-\log f_{Z_1}(z)}{z^d} \geq C_1 - \epsilon$ and $\frac{-\log f_{Z_2}(z)}{z^d} \geq C_2 - \epsilon$ for $z \geq M$.

Consider $Z > \max(\frac{1}{p}, \frac{1}{1-p}) \cdot M$,

$$\begin{aligned}
f_Z(z) &= \int_{-\infty}^{\infty} f_{Z_1}(z_1) f_{Z_2}(z - z_1) dz_1 \\
&= \int_{-\infty}^{M} f_{Z_1}(z_1) f_{Z_2}(z - z_1) dz_1 + \int_{M}^{pz} f_{Z_1}(z_1) f_{Z_2}(z - z_1) dz_1 \\
&\quad + \int_{pz}^{\infty} f_{Z_1}(z_1) f_{Z_2}(z - z_1) dz_1 \\
&\leq \int_{-\infty}^{M} e^{-(C_2-\epsilon)(z-z_1)^d} f_{Z_1}(z_1) dz_1 + \int_{M}^{pz} e^{-(C_1-\epsilon)z_1^d} e^{-(C_2-\epsilon)(z-z_1)^d} dz_1 \\
&\quad + \int_{pz}^{\infty} e^{-(C_1-\epsilon)z_1^d} f_{Z_2}(z - z_1) dz_1 \\
&\leq e^{-(C_2-\epsilon)(z-M)^d} + \int_{M}^{pz} e^{-z^d(C_1(z_1/z)^d + C_2(1-z_1/z)^d) + \epsilon O(z^d)} dz_1 + e^{-(C_1-\epsilon)p^d z^d} \\
&\leq e^{-C_2(z-M)^d + \epsilon O(z^d)} + e^{-z^d(\min_{0\leq u\leq p} C_1 u^d + C_2(1-u)^d) + \epsilon O(z^d)} + e^{-C_1 p^d z^d + \epsilon O(z^d)} \\
&\leq 3e^{-z^d \min(C_2, \min_{0\leq u\leq p} C_1 u^d + C_2(1-u)^d, C_1 p^d) + \epsilon O(z^d)}
\end{aligned}$$

$$\Rightarrow \quad \frac{-\log f_Z(z)}{z^d} \geq \frac{-\log 3}{z^d} + \min(C_2, \min_{0\leq u\leq p} C_1 u^d + C_2(1-u)^d, C_1 p^d) - \epsilon O(1)$$

$$\Rightarrow \quad \liminf_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \geq \min(\min_{0\leq u\leq p} C_1 u^d + C_2(1-u)^d), C_1 p^d) - \epsilon O(1)$$

Since ϵ can be arbitrarily small, $\liminf_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \geq \min(\min_{0\leq u\leq p} C_1u^d + C_2(1-u)^d, C_1p^d)$ for $\forall\ 0 < p < 1$, so $\liminf_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \geq \max_{0<p<1} \min(\min_{0\leq u\leq p} C_1u^d + C_2(1-u)^d, C_1p^d)$. By the continuity of function $\min(\min_{0\leq u\leq p} C_1u^d+C_2(1-u)^d, C_1p^d)$, we have $\liminf_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \geq \max_{0\leq p\leq 1} \min(\min_{0\leq u\leq p} C_1u^d + C_2(1-u)^d, C_1p^d)$.

Now let's consider the upper bound. $\forall\,\epsilon > 0, \forall\ 0 < q < 1, \exists N(\epsilon, q) > 0, s.t\ \frac{-\log f_{Z_1}(z)}{z^d} \leq C_1 + \epsilon$, $\frac{-\log f_{Z_2}(z)}{z^d} \leq C_2 + \epsilon$ for $z \geq N$ and $P(0 \leq Z_1 \leq N) \geq \frac{1}{2}P(Z_1 \geq 0), P(0 \leq Z_2 \leq N) \geq \frac{1}{2}P(Z_2 \geq 0)$.

Consider $Z > \max(\frac{2}{q}, \frac{1}{1-q}) \cdot N$,

$$\begin{aligned}
f_Z(z) &= \int_{-\infty}^{\infty} f_{Z_1}(z_1)f_{Z_2}(z-z_1)dz_1 \\
&= \int_{-\infty}^{qz-N} f_{Z_1}(z_1)f_{Z_2}(z-z_1)dz_1 + \int_{qz-N}^{qz} f_{Z_1}(z_1)f_{Z_2}(z-z_1)dz_1 \\
&\quad + \int_{qz}^{\infty} f_{Z_1}(z_1)f_{Z_2}(z-z_1)dz_1 \\
&\geq \int_{-N}^{qz-N} f_{Z_1}(z_1)e^{-(C_2+\epsilon)(z-z_1)^d}dz_1 + \int_{qz-N}^{qz} e^{-(C_1+\epsilon)z_1^d}e^{-(C_2+\epsilon)(z-z_1)^d}dz_1 \\
&\quad + \int_{qz}^{z} e^{-(C_1+\epsilon)z_1^d}f_{Z_2}(z-z_1)dz_1 \\
&\geq \frac{1}{2}P(Z_1 \geq 0)e^{-(C_2+\epsilon)(z+N)^d} + e^{-(C_1+\epsilon)q^dz^d}e^{-(C_2+\epsilon)(z-qz+N)^d}N \\
&\quad + \frac{1}{2}P(Z_2 \geq 0)e^{-(C_1+\epsilon)z^d} \\
&\geq K \cdot e^{-z^d\min(C_2, C_1q^d+C_2(1-q)^d, C_1)-\epsilon O(z^d)} \qquad \text{(where K is some constant)} \\
\Rightarrow\ & \frac{-\log f_Z(z)}{z^d} \leq \frac{-\log K}{z^d} + \min(C_2, C_1q^d + C_2(1-q)^d, C_1) + \epsilon O(1)
\end{aligned}$$

Since ϵ can be arbitrarily small, this implies $\limsup_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \leq \min(C_2, C_1q^d + C_2(1-q)^d, C_1)$. Also, since it holds for any $0 < q < 1$, $\limsup_{z\to\infty} \frac{-\log f_Z(z)}{z^d} \leq \min_{0<q<1} \min(C_2, C_1q^d + C_2(1-q)^d, C_1) = \min_{0\leq q\leq 1}(C_1q^d + C_2(1-q)^d)$.

Finally we need to show the upper bound of the limsup equals to the lower bound of the liminf, i.e., $\max_{0\le p\le 1}\min(\min_{0\le u\le p} C_1u^d + C_2(1-u)^d, C_1p^d) = \min_{0\le q\le 1}(C_1q^d + C_2(1-q)^d)$.

Consider the LHS, $\min_{0\le u\le p} C_1u^d + C_2(1-u)^d$ is a decreasing function of p, ranging from $\min_{0\le u\le 1} C_1u^d + C_2(1-u)^d$ to C_2 and C_1p^d is a strictly increasing function of p ranging from 0 to C_1, they have a unique crossing at p^* because $0 < \min_{0\le u\le 1} C_1u^d + C_2(1-u)^d \le C_1 \le C_2$. And $LHS = \min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = C_1(p^*)^d$. Keeping this equality in mind, we consider the three cases below.

Case 1: $d > 1$.
$C_1u^d + C_2(1-u)^d$ is convex, assume $0 \le u^* \le 1$ achieves the minimal value. Then if $p^* < u^*$, $\min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = C_1(p^*)^d + C_2(1-p^*)^d \ne C_1(p^*)^d$, contradiction. So $p^* \ge u^*$, this implies $\min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = \min_{0\le u\le 1} C_1u^d + C_2(1-u)^d$, i.e., LHS=RHS.

Case 2: $d = 1$.
LHS=$\min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = \min_{0\le u\le p^*}(C_1 - C_2)u + C_2 = (C_1 - C_2)p^* + C_2$. $(C_1 - C_2)p^* + C_2 = C_1p^* \Rightarrow p^* = 1$. So LHS=RHS.

Case 3: $0 < d < 1$.
$C_1u^d + C_2(1-u)^d$ is concave, assume $0 \le u^* \le 1$ achieves the maximal value. If $p^* \le u^*$, $\min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = C_2 \ne C_1(p^*)^d$ in general; if $p^* > u^*$, $\min_{0\le u\le p^*} C_1u^d + C_2(1-u)^d = \min(C_2, C_1(p^*)^d + C_2(1-p^*)^d)$, since this equals $C_1(p^*)^d$, we get $p^* = 1$. So LHS=RHS.

Combining all the analysis above, we know $\lim_{z\to\infty} \frac{\log f_Z(z)}{z^d}$ exists and equals to $\min_{0\le u\le 1} C_1u^d + C_2(1-u)^d$. When $0 < d \le 1$, the limit C is $\min(C_1, C_2)$, when

$d > 1$, the limit C is $C_1(\frac{C_2^{1/(d-1)}}{C_1^{1/(d-1)}+C_2^{1/(d-1)}})^d + C_2(\frac{C_1^{1/(d-1)}}{C_1^{1/(d-1)}+C_2^{1/(d-1)}})^d$.

Similiar results hold when $z \to -\infty$.

We have proved the main theorem is valid in a very general setting. This provides a guideline to achieve efficient discrimination by implementing a sequential testing scheme. In the following chapter, we will demonstrate the main theorem only implicitly (in §4.4.4), due to no explicit form for σ in our implementation, and not even mention $\sigma \to 0$.

3.5 Appendix

Lemma 1: $\forall m > 0, E|Z|^m < \infty$, $E|\log f(Z)|^m < \infty$.

Proof: Choose $N_1 > 0$ such that $|\phi(z) - a| < \frac{a}{2}$ for $z > N_1$ and $|\phi(z) - b| < \frac{b}{2}$ for $z < -N_1$.

$$\begin{aligned} E|Z|^m &= \int_{-\infty}^{\infty} |z|^m f(z)dz \\ &= \int_{N_1}^{\infty} |z|^m f(z)dz + \int_{-N_1}^{N_1} |z|^m f(z)dz + \int_{-\infty}^{-N_1} |z|^m f(z)dz \\ &\leq N_1^m + \int_{N_1}^{\infty} |z|^m e^{-\frac{a}{2}|z|^d} dz + \int_{-\infty}^{-N_1} |z|^m e^{-\frac{b}{2}|z|^{d'}} dz \\ &< \infty \end{aligned}$$

Choose $N_2 > 0$ such that $|\phi(z) - a| < 1$ for $z > N_2$ and $|\phi(z) - b| < 1$ for $z < -N_2$.

$$
\begin{aligned}
E|\log f(Z)|^m &= \int_{-\infty}^{\infty} |\log f(z)|^m f(z)dz \\
&= \int_{N_2}^{\infty} |\log f(z)|^m f(z)dz + \int_{-N_2}^{N_2} |\log f(z)|^m f(z)dz \\
&\quad + \int_{-\infty}^{-N_2} |\log f(z)|^m f(z)dz \\
&\leq \max_{-N_2 \leq z \leq N_2} |\log f(z)|^m + \int_{N_2}^{\infty} ((a+1)|z|^d)^m f(z)dz \\
&\quad + \int_{-\infty}^{-N_2} ((b+1)|z|^{d'})^m f(z)dz \\
&\leq \max_{-N_2 \leq z \leq N_2} |\log f(z)|^m + (a+1)^m E|Z|^{md} + (b+1)^m E|Z|^{md'} \\
&\overset{(1)}{<} \infty
\end{aligned}
$$

Inequality (1) holds because $|\log f(z)|^m$ is uniformly continuous between $[-N_2, N_2]$. Especially $E|\log f(Z)| < \infty, var(\log f(Z)) < \infty$.

Lemma 2: $\lim_{\sigma\to 0} \frac{\mu_1}{a\sigma^{-d}} = 1$, $\lim_{\sigma\to 0} \frac{S_1}{\sigma^{-d}} = 0$.

Proof: Given $\lim_{z\to\infty} \phi(z) = a$, $\lim_{z\to-\infty} \phi(z) = b$
$\Rightarrow \exists M > 0,\ s.t\ \forall |z| > M, |\phi(z)| \leq \alpha \doteq \max(a,b) + 1$, i.e., $|-\log f(z)| \leq \alpha|z|^d$ for $|z| > M$.

Let $\beta \doteq \max_{|z|\leq M} |-\log f(z)|$, we have $|-\log f(z)| \leq \alpha|z|^d + \beta$ for all z.

Therefore we have $\sigma^d|-\log f(Z + \frac{1}{\sigma})| \leq \sigma^d(\alpha|Z + \frac{1}{\sigma}|^d + \beta) = \alpha|1 + \sigma Z|^d + \sigma^d\beta$ $\leq \alpha(1 + |Z|)^d + \beta$ (when $\sigma < 1$), which is integrable by Lemma 1.

$$\begin{aligned}
\lim_{\sigma\to 0}\frac{\mu_1}{a\sigma^{-d}} &= \lim_{\sigma\to 0}\frac{E(\log f(Z_1)-\log f(Z_1+\frac{1}{\sigma}))}{a\sigma^{-d}}\\
&= \lim_{\sigma\to 0}\frac{E(\sigma^d(-\log f(Z_1+\frac{1}{\sigma})))}{a}\\
&\overset{(2)}{=} \frac{1}{a}E\lim_{\sigma\to 0}(\sigma^d\frac{-\log f(Z_1+\frac{1}{\sigma})}{|Z_1+\frac{1}{\sigma}|^d}|Z_1+\frac{1}{\sigma}|^d)\\
&= 1
\end{aligned}$$

$$\begin{aligned}
\lim_{\sigma\to 0}\frac{S_1^2}{\sigma^{-2d}} &= var(\sigma^d(\log f(Z_1)-\log f(Z_1+\frac{1}{\sigma})))\\
&\le \lim_{\sigma\to 0}2(var(\sigma^d\log f(Z_1))+var(\sigma^d(-\log f(Z_1+\frac{1}{\sigma}))))\\
&= 2\lim_{\sigma\to 0}E[(\sigma^d(-\log f(Z_1+\frac{1}{\sigma})))^2]-[E(\sigma^d(-\log f(Z_1+\frac{1}{\sigma})))]^2\\
&\overset{(3)}{=} 2(E[(\lim_{\sigma\to 0}\sigma^d\frac{-\log f(Z_1+\frac{1}{\sigma})}{|Z_1+\frac{1}{\sigma}|^d}|Z_1+\frac{1}{\sigma}|^d)^2]-a^2)\\
&= 2(a^2-a^2)\\
&= 0
\end{aligned}$$

So $\lim_{\sigma\to 0}\frac{S_1}{\sigma^{-d}}=0$.

Equality (2) and (3) hold by DCT.

Lemma 3: *Let W_n be a sequence of continuous RVs with mean 0 and variance 1, for any fixed r $(0<r<1)$, consider a sequence of η_n such that $P(W_n\ge\eta_n)=r$, then $|\eta_n|\le max(\frac{1}{\sqrt{r}},\frac{1}{\sqrt{1-r}})$ for any n.*

Proof: Fix n, For $\eta_n \geq 0$,

$$\begin{aligned} P(W_n \geq \eta_n) &= r \\ &\leq P(|W_n|^2 \geq \eta_n^2) \\ &\leq \frac{E(W_n^2)}{\eta_n^2} = \frac{1}{\eta_n^2} \quad \textit{since } E(W_n^2) = 1 \\ &\Rightarrow |\eta_n| \leq \frac{1}{\sqrt{r}} \end{aligned}$$

For $\eta_n < 0$, $P(W_n \geq \eta_n) = r \Rightarrow P(W_n < \eta_n) = 1 - r$,

$$\begin{aligned} P(W_n < \eta_n) &= 1 - r \\ &\leq P(|W_n|^2 \geq \eta_n^2) \\ &\leq \frac{E(W_n^2)}{\eta_n^2} = \frac{1}{\eta_n^2} \\ &\Rightarrow |\eta_n| \leq \frac{1}{\sqrt{1-r}} \end{aligned}$$

Therefore $|\eta_n| \leq max(\frac{1}{\sqrt{r}}, \frac{1}{\sqrt{1-r}})$ for all n.

Lemma 4: $\lim_{k\to\infty} \frac{P(Z\geq k)}{e^{-ak^d+o(k^d)}} = 1$*, for* $d > 0$.

Proof:

$$\begin{aligned} \lim_{k\to\infty} \frac{\int_k^\infty e^{-ax^d}dx}{k^{1-d}e^{-ak^d}} &\overset{(4)}{=} \lim_{k\to\infty} \frac{-e^{-ak^d}}{(1-d)k^{-d}e^{-ak^d} + k^{1-d}e^{-ak^d}(-a)dk^{d-1}} \\ &= \lim_{k\to\infty} \frac{1}{(d-1)k^{-d} + ad} \\ &= \frac{1}{ad} \end{aligned}$$

Equality (4) comes from taking the derivative of k on both sides.

$\forall\epsilon > 0, \exists K > 0, s.t\ \forall k > K,\ \int_k^\infty e^{-(a+\epsilon)x^d}dx \leq P(Z \geq k) = \int_k^\infty e^{-\phi(x)x^d}dx \leq$

$\int_k^\infty e^{-(a-\epsilon)x^d}dx$

We know the upper and lower bounds are of the order $k^{1-d}e^{-(a-\epsilon)k^d}$ and $k^{1-d}e^{-(a+\epsilon)k^d}$ respectively. Since ϵ can be arbitrary small, the lemma follows.

Lemma 5: $\lim_{\sigma\to 0}\frac{P(\frac{u}{\sigma}\le Z_1\le\frac{1}{\sigma},\frac{v}{\sigma}\le Z_2\le\frac{1}{\sigma}-L)}{P(Z_1\ge\frac{u}{\sigma},Z_2\ge\frac{v}{\sigma})} = 1$, *where* $0 < u,v < 1$ *and* $L > 0$ *is any constant.*

Proof: It is sufficient to show $\lim_{\sigma\to 0}\frac{P(Z_1>\frac{1}{\sigma},Z_2>\frac{1}{\sigma}-L)}{P(Z_1\ge\frac{u}{\sigma},Z_2\ge\frac{v}{\sigma})} = 0$.
Choose $\epsilon = \frac{a}{2}\cdot\frac{2-(u^d+v^d)}{2}$.

$$\begin{aligned}
\lim_{\sigma\to 0}\frac{P(Z_1>\frac{1}{\sigma},Z_2>\frac{1}{\sigma}-L)}{P(Z_1\ge\frac{u}{\sigma},Z_2\ge\frac{v}{\sigma})} &= \lim_{\sigma\to 0}\frac{P(Z_1>\frac{1}{\sigma})P(Z_2>\frac{1}{\sigma}-L)}{P(Z_1\ge\frac{u}{\sigma})P(Z_2\ge\frac{v}{\sigma})} \\
&= \lim_{\sigma\to 0}\frac{e^{-a\sigma^{-d}-a(1-\sigma L)^d\sigma^{-d}+o(\sigma^{-d})}}{e^{-au^d\sigma^{-d}-av^d\sigma^{-d}+o(\sigma^{-d})}} \\
&\overset{(5)}{\le} \lim_{\sigma\to 0}e^{\sigma^{-d}(au^d+av^d-a-a(1-\sigma L)^d+2\epsilon)} \\
&= \lim_{\sigma\to 0}e^{a\sigma^{-d}(\frac{u^d+v^d}{2}-(1-\sigma L)^d)} \\
&= 0
\end{aligned}$$

Inequality (5) holds because $|o(\sigma^{-d})|\le\epsilon\sigma^{-d}$ when σ is small enough.

Lemma 6: *Equation* $-u^d+(1+u)^d+v^d-\frac{b}{a}(1-v)^d = 2$ *has a solution* u^*,v^* *such that* $u^*>0, v^*>0$ *and* $(u^*)^d+(v^*)^d<1$.

Proof: Let $F(u,v)\doteq -u^d+(1+u)^d+v^d-\frac{b}{a}(1-v)^d-2$, $\bar{F}(u,v)\doteq u^d+v^d-1$. $F(0,1)=\bar{F}(0,1)=0$.

The implicit functions $v(u),\bar{v}(u)$ from $F(u,v)=0,\bar{F}(u,v)=0$ are decreasing. Because by implicit function theorem,

$$\begin{aligned}\frac{dv}{du} &= -\frac{d\cdot(1+u)^{d-1} - d\cdot u^{d-1}}{d\cdot b/a\cdot(1-v)^{d-1} + d\cdot v^{d-1}} \le 0\\ \frac{d\bar{v}}{du} &= -\frac{d\cdot u^{d-1}}{d\cdot v^{d-1}} \le 0\end{aligned}$$

Note: $\frac{dv}{du}|_{(0,1)} = -1$ and $\frac{d\bar{v}}{du}|_{(0,1)} = 0$. Geometrically it implies that curve $F(u,v)=0$ dips into the interior of the region $u^d + v^d \le 1$ around point (0,1). Therefore the lemma holds.

Lemma 7: *There exists $\epsilon_M > 0$, such that $\forall \epsilon < \epsilon_M, h_\epsilon(u,v) \doteq (1+u)^d - \frac{a+\epsilon}{a-\epsilon}u^d + v^d - \frac{b+\epsilon}{a-\epsilon}(1-v)^d$ is an increasing function of u, v, where $u_0 \le u \le 1, v_0 \le v \le 1$ for $\forall 0 < u_0, v_0 < 1$.*

Proof: It's easy to see $h_\epsilon(u,v)$ is an increasing function of v for $v_0 \le v \le 1$. It is sufficient to show $\frac{\partial h_\epsilon(u,v)}{\partial u} > 0$ for small ϵ.

$$\frac{\partial h_\epsilon(u,v)}{\partial u} = d(1+u)^{d-1} - d\frac{a+\epsilon}{a-\epsilon}u^{d-1} > 0 \Rightarrow \epsilon < \frac{a((1+u)^{d-1} - u^{d-1})}{(1+u)^{d-1} + u^{d-1}}$$

Let $\epsilon_M = \frac{a((1+u_0)^{d-1}-1)}{2^{d-1}+1}$, then for $\forall \epsilon < \epsilon_M, h_\epsilon(u,v)$ is an increasing function of u, v.

Lemma 8: $\lim_{\sigma\to 0} \frac{P(r_1 - r_3 \ge 2\mu_1 + \sqrt{2}\alpha_r S_1)}{P(Z_1 \ge \frac{u_\delta}{\sigma}, Z_2 \ge \frac{v_\delta}{\sigma})} \ge 1$*, for some $u_\delta > 0, v_\delta > 0$ such that $u_\delta^d + v_\delta^d < 1$.*

Proof: The general idea is to find a lower bound of $P(r_1 - r_3 \ge 2\mu_1 + \sqrt{2}\alpha_r S_1)$, which is bigger than $P(Z_1 \ge \frac{u_\delta}{\sigma}, Z_2 \ge \frac{v_\delta}{\sigma})$, using ϵ argument.

Let $g_1(Z_1, Z_2) \doteq -\phi(Z_1)|Z_1|^d + \phi(Z_1 + \frac{1}{\sigma})|Z_1 + \frac{1}{\sigma}|^d + \phi(Z_2)|Z_2|^d - \phi(Z_2 - \frac{1}{\sigma})|Z_2 - \frac{1}{\sigma}|^d$, i.e., $g_1(Z_1, Z_2) = r_1 - r_3$ when ϕ is defined.

We start with equation $h(u,v) = -u^d + (1+u)^d + v^d - \frac{b}{a}(1-v)^d = 2$. By Lemma 6, there exists a solution u^*, v^* such that $u^* > 0, v^* > 0$ and $(u^*)^d + (v^*)^d < 1$.

Since (u^*, v^*) is an interior point of open region $S \doteq \{(u,v) : u > 0, v > 0, u^d + v^d < 1\}$, there exists a neighborhood $B_{(u^*,v^*)}(\delta)$, such that $B_{(u^*,v^*)}(\delta) \subset S$.

Let $u_\delta \doteq u^* + \delta, v_\delta \doteq v^* + \delta$, $\delta > 0$, we know $u_\delta^d + v_\delta^d < 1$.

Since $h(u,v)$ is an increasing function of u, v, when $0 \le u \le 1, 0 \le v \le 1$, and $h(u^*, v^*) = 2$, so $h(u_\delta, v_\delta) > 2$.

Let $\epsilon_\delta \doteq \frac{a(h(u_\delta,v_\delta)-2)}{2^d+2a+\sqrt{2}|\alpha_r|+3}$; for $\forall \epsilon \le \epsilon_\delta$, $h(u_\delta, v_\delta) \ge 2 + \epsilon \frac{2^d+2a+\sqrt{2}|\alpha_r|+3}{a}$

$$
\begin{aligned}
\Leftrightarrow\quad & ah(u_\delta, v_\delta) \ge 2a + \epsilon(2^d + 2a + \sqrt{2}|\alpha_r| + 3) \\
\Leftrightarrow\quad & [a(1+u_\delta)^d - \epsilon 2^d] + [-au_\delta^d - \epsilon] + [av_\delta^d - \epsilon] + [-b(1-v_\delta)^d - \epsilon] \\
& \ge 2a(1+\epsilon) + \epsilon\sqrt{2}|\alpha_r| \\
\Rightarrow\quad & (a-\epsilon)(1+u_\delta)^d - (a+\epsilon)u_\delta^d + (a-\epsilon)v_\delta^d - (b+\epsilon)(1-v_\delta)^d \\
& \ge 2a(1+\epsilon) + \epsilon\sqrt{2}\alpha_r \\
\Leftrightarrow\quad & (a-\epsilon)h_\epsilon(u_\delta, v_\delta) \ge 2a + \epsilon(2a + \sqrt{2}\alpha_r)
\end{aligned}
$$

So $\forall \epsilon < \min(\epsilon_\delta, \epsilon_M), (a-\epsilon)h_\epsilon(u,v) \ge 2a + \epsilon(2a + \sqrt{2}\alpha_r)$, for $u_\delta \le u \le 1, v_\delta \le v \le 1$ (By Lemma 7).

Let $\epsilon_0 \doteq \frac{1}{2}\min(\epsilon_\delta, \epsilon_M)$.
Consider $\frac{u_\delta}{\sigma} \le Z_1 \le \frac{1}{\sigma}, \frac{v_\delta}{\sigma} \le Z_2 \le \frac{1}{\sigma} - L_{\epsilon_0}$, where L_{ϵ_0} is a constant such that $|\phi(x) - b| \le \epsilon_0$ for $x \le -L_{\epsilon_0}$.

$\exists\sigma_1 > 0, s.t\ |\phi(Z_1) - a| \leq \epsilon_0$ for $\sigma < \sigma_1$.

$\exists\sigma_2 > 0, s.t\ |\phi(Z_2) - a| \leq \epsilon_0$ for $\sigma < \sigma_2$.

$\exists\sigma_3 > 0, s.t\ |\frac{\mu_1}{a\sigma^{-d}} - 1| \leq \epsilon_0$ for $\sigma < \sigma_3$.

$\exists\sigma_4 > 0, s.t\ |\frac{S_1}{\sigma^{-d}}| \leq \epsilon_0$ for $\sigma < \sigma_4$.

Consider $\sigma < \min(\frac{1-v_\delta}{L_{\epsilon_0}}, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$, $\{\frac{u_\delta}{\sigma} \leq Z_1 \leq \frac{1}{\sigma}, \frac{v_\delta}{\sigma} \leq Z_2 \leq \frac{1}{\sigma} - L_{\epsilon_0}\}$

$$\begin{aligned}
&\subseteq \{(a-\epsilon_0)h_{\epsilon_0}(\sigma Z_1, \sigma Z_2) \geq 2a + \epsilon_0(2a + \sqrt{2}\alpha_r)\} \\
&\subseteq \{\sigma^d g_1(Z_1, Z_2) \geq \sigma^d(2\mu_1 + \sqrt{2}\alpha_r S_1)\} \\
&\subseteq \{g_1(Z_1, Z_2) \geq 2\mu_1 + \sqrt{2}\alpha_r S_1\}
\end{aligned}$$

Equivalently, $\lim_{\sigma\to 0} \frac{P(r_1 - r_3 \geq 2\mu_1 + \sqrt{2}\alpha_r S_1)}{P(\frac{u_\delta}{\sigma} \leq Z_1 \leq \frac{1}{\sigma}, \frac{v_\delta}{\sigma} \leq Z_2 \leq \frac{1}{\sigma} - L_{\epsilon_0})} \geq 1$.

Combining with $\lim_{\sigma\to 0} \frac{P(\frac{u_\delta}{\sigma} \leq Z_1 \leq \frac{1}{\sigma}, \frac{v_\delta}{\sigma} \leq Z_2 \leq \frac{1}{\sigma} - L_{\epsilon_0})}{P(Z_1 \geq \frac{u_\delta}{\sigma}, Z_2 \geq \frac{v_\delta}{\sigma})} = 1$ (by Lemma 5), we get $\lim_{\sigma\to 0} \frac{P(r_1 - r_3 \geq 2\mu_1 + \sqrt{2}\alpha_r S_1)}{P(Z_1 \geq \frac{u_\delta}{\sigma}, Z_2 \geq \frac{v_\delta}{\sigma})} \geq 1$.

Lemma 9: $\lim_{\sigma\to 0} \frac{P(-r_3 \geq \mu_1 + \beta_{\sqrt{r}} S_1)}{P(Z_2 \geq \frac{1-\eta}{\sigma})} \leq 1$ *for any* $\eta > 0$.

Proof: The general idea is to find a lower bound of $P(-r_3 < \mu_1 + \beta_{\sqrt{r}} S_1)$, which is bigger than $P(Z_2 < \frac{1-\eta}{\sigma})$, using ϵ argument.

let $g_2(Z) \doteq -\phi(Z - \frac{1}{\sigma})|Z - \frac{1}{\sigma}|^d + \phi(Z)|Z|^d$, i.e., $g_2(Z_2) = -r_3$ for $Z_2 \neq 0,\ \frac{1}{\sigma}$.

We will show $\{-r_3 \geq \mu_1 + \beta_{\sqrt{r}} S_1\} \subseteq \{Z_2 \geq \frac{1-\eta}{\sigma}\}$ when σ is small enough. Since when $Z_2 = 0$, event $\{-r_3 = \log f(-\frac{1}{\sigma}) - \log f(0) \geq \mu_1 + \beta_{\sqrt{r}} S_1\}$ is empty for small σ and when $Z_2 = \frac{1}{\sigma}$, event $\{Z_2 \geq \frac{1-\eta}{\sigma}\}$ is the whole space and for Z_2 take other values, $-r_3 = g_2(Z_2)$, therefore we only need to show $\{g_2(Z_2) \geq \mu_1 + \beta_{\sqrt{r}} S_1\} \subseteq \{Z_2 \geq \frac{1-\eta}{\sigma}\}$ for small σ. Equivalently, to show $\{Z_2 < \frac{1-\eta}{\sigma}\} \subseteq \{g_2(Z_2) < \mu_1 + \beta_{\sqrt{r}} S_1\}$.

Fix $1 > \eta > 0$, let $\epsilon_1 \doteq \min(\frac{a(1-(1-\eta)^d)}{2((1-\eta)^d+a+|\beta_{\sqrt{r}}|)}, a, b)$.
Rewrite $g_2(Z_2) = -\phi(Z_2-\frac{1}{\sigma})|Z_2-\frac{1}{\sigma}|^d+(-\log f(Z_2))1_{\{|Z_2|\leq K_{\epsilon_1}\}}+\phi(Z_2)|Z_2|^d 1_{\{|Z_2|>K_{\epsilon_1}\}}$ where K_{ϵ_1} is a constant such that $|\phi(x)-a| < \epsilon_1$ for $x > K_{\epsilon_1}$ and $|\phi(x)-b| < \epsilon_1$ for $x < -K_{\epsilon_1}$.

Given $Z_2 < \frac{1-\eta}{\sigma}$,
$\exists \sigma_5 > 0, s.t\ |\phi(Z_2-\frac{1}{\sigma})-b| \leq \epsilon_1$ for $\sigma < \sigma_5$.
$\exists \sigma_6 > 0, s.t\ |\frac{\mu_1}{a\sigma^{-d}}-1| \leq \epsilon_1$ for $\sigma < \sigma_6$.
$\exists \sigma_7 > 0, s.t\ |\frac{S_1}{\sigma^{-d}}| \leq \epsilon_1$ for $\sigma < \sigma_7$.

Consider the upper bound of $g_2(Z)$; let $H_{\epsilon_1}(Z) \doteq (-b+\epsilon_1)|Z-\frac{1}{\sigma}|^d-(\log f(Z))1_{\{|Z|\leq K_{\epsilon_1}\}}$ $+(a+\epsilon_1)|Z|^d 1_{\{Z>K_{\epsilon_1}\}} + (b+\epsilon_1)|Z|^d 1_{\{Z<-K_{\epsilon_1}\}}$
Event $\{H_{\epsilon_1}(Z_2) < a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}\} \subseteq \{g_2(Z_2) < \mu_1 + \beta_{\sqrt{r}}S_1\}$ for $\sigma < \min(\sigma_5, \sigma_6, \sigma_7)$.

Now the only thing we need to show is $H_{\epsilon_1}(Z) < a\sigma^{-d}(1-\epsilon_1)-\epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}$ for $Z < \frac{1-\eta}{\sigma}$, when σ is small enough, i.e., $\{Z < \frac{1-\eta}{\sigma}\} \subseteq \{H_{\epsilon_1}(Z) < a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}\}$.

Let $M_{\epsilon_1} \doteq \max_{-K_{\epsilon_1}\leq z\leq K_{\epsilon_1}} |-\log f(z)|$, consider $\sigma < \min(\sigma_5, \sigma_6, \sigma_7, (\frac{a-(a+|\beta_{\sqrt{r}}|)\epsilon_1}{M_{\epsilon_1}})^{1/d})$

Case 1: $K_{\epsilon_1} < Z < \frac{1-\eta}{\sigma}$, $H_{\epsilon_1}(Z) \leq (a+\epsilon_1)|Z|^d < a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}$. (by definition of ϵ_1)

Case 2: $-K_{\epsilon_1} \leq Z \leq K_{\epsilon_1}$, $H_{\epsilon_1}(Z) \leq -\log f(Z) \leq M_{\epsilon_1} \leq a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}$.

Case 3: $Z < -K_{\epsilon_1}$, $H_{\epsilon_1}(Z) = (-b+\epsilon_1)|Z-\frac{1}{\sigma}|^d + (b+\epsilon_1)|Z|^d$. $\exists \sigma_8 > 0, s.t\ H_{\epsilon_1}(Z) \leq 0 \leq a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}$ for $\sigma < \sigma_8$.

Therefore, we reach the conclusion that when $\sigma < \min(\sigma_5, \sigma_6, \sigma_7, \sigma_8, (\frac{a-(a+|\beta_{\sqrt{r}}|)\epsilon_1}{M_{\epsilon_1}})^{1/d})$, $\{Z < \frac{1-\eta}{\sigma}\} \subseteq \{H_{\epsilon_1}(Z) < a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}\}$.

So $\lim_{\sigma\to 0} \frac{P(-r_3 \geq \mu_1 + \beta_{\sqrt{r}} S_1)}{P(Z_2 \geq \frac{1-\eta}{\sigma})} \leq 1$ for any $\eta > 0$.

Lemma 10: *When the left tail decays at rate $0 < d' \leq d$, all the previous lemmas hold and all the arguments follow.*

Proof: In the generalized case, all the previous lemma hold. The related lemmas are Lemma 2, 8, 9.

Let $\phi(z) \doteq \frac{-\log(f_(z))}{|z|^d}$ for $z > 0$ and $\phi(z) \doteq \frac{-\log(f_(z))}{|z|^{d'}}$ for $z < 0$

$f(z) = e^{-\phi(z)z^d} 1_{\{z>0\}} + e^{-\phi(z)z^{d'}} 1_{\{z<0\}} + f(0)1_{\{z=0\}}$

Consider the proof of Lemma 2; all we need is to find an intergrable dominate function, so we can apply DCT. Since $d' \leq d$, the previous dominate function is still valid. In fact, for any $d' > 0$, lemma 2 holds by DCT.

Consider the proof of Lemma 8; the only changes are as follows.
Given $\frac{u_\delta}{\sigma} \leq Z_1 \leq \frac{1}{\sigma}, \frac{v_\delta}{\sigma} \leq Z_2 \leq \frac{1}{\sigma} - L_{\epsilon_0}$,

$g_1(Z_1, Z_2) \doteq -\phi(Z_1)|Z_1|^d + \phi(Z_1 + \frac{1}{\sigma})|Z_1 + \frac{1}{\sigma}|^d + \phi(Z_2)|Z_2|^d - \phi(Z_2 - \frac{1}{\sigma})|Z_2 - \frac{1}{\sigma}|^{d'}$

$h(u, v) \doteq -u^d + (1+u)^d + v^d - \frac{b(1-v)^{d'}}{a}\sigma^{d-d'}$

$h_\epsilon(u, v) \doteq (1+u)^d - \frac{a+\epsilon}{a-\epsilon}u^d + v^d - \frac{b+\epsilon}{a-\epsilon}(1-v)^{d'}\sigma^{d-d'}$

$h(u, v), h_\epsilon(u, v)$ share all the property as before, except that they are getting bigger when σ is small, which is in our favor. The exactly same argument follows.

Consider the proof of Lemma 9; the only changes are as follows.

Given $Z_2 < \frac{1-\eta}{\sigma}$,

$g_2(Z_2) \doteq -\phi(Z_2 - \frac{1}{\sigma})|Z_2 - \frac{1}{\sigma}|^{d'} + (-\log f(Z_2))1_{\{|Z_2| \leq K_{\epsilon_1}\}} + \phi(Z_2)|Z_2|^d 1_{\{Z_2 > K_{\epsilon_1}\}} + \phi(Z_2)|Z_2|^{d'} 1_{\{Z_2 < -K_{\epsilon_1}\}}$

$H_{\epsilon_1}(Z) \doteq (-b+\epsilon_1)|Z - \frac{1}{\sigma}|^{d'} + (-\log f(Z))1_{\{|Z| \leq K_{\epsilon_1}\}} + (a+\epsilon_1)|Z|^d 1_{\{Z > K_{\epsilon_1}\}} + (b+\epsilon_1)|Z|^{d'} 1_{\{Z < -K_{\epsilon_1}\}}$

Changes to case 3 in Lemma 9: when $Z < -K_{\epsilon_1}$, $H_{\epsilon_1}(Z) = (-b+\epsilon_1)|Z - \frac{1}{\sigma}|^{d'} + (b+\epsilon_1)|Z|^{d'}$. $\exists \sigma_8^{'} > 0, s.t\ H_{\epsilon_1}(Z) \leq 0 \leq a\sigma^{-d}(1-\epsilon_1) - \epsilon_1|\beta_{\sqrt{r}}|\sigma^{-d}$ for $\sigma < \sigma_8^{'}$.

After the changes, we come to the same conclusion. In fact, lemma 9 holds for any $d' > 0$.

Lemma 11: *When Z_1, Z_2 are not identical but independent, the theorem holds.*

Proof: A few more notations are defined below.

$\tilde{r}_3 \doteq \log \frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1 - \frac{1}{\sigma})}$

$\tilde{\mu}_3 \doteq E(\tilde{r}_3)$ $\tilde{S}_3 \doteq s.d(\tilde{r}_3)$

$\phi(z) \doteq \frac{-\log(f_{Z_1}(z))}{|z|^d}$ for $z > 0$ and $\phi(z) \doteq \frac{-\log(f_{Z_1}(z))}{|z|^{d'}}$ for $z < 0$

$\psi(z) \doteq \frac{-\log(f_{Z_2}(z))}{|z|^d}$ for $z > 0$ and $\psi(z) \doteq \frac{-\log(f_{Z_2}(z))}{|z|^{d'}}$ for $z < 0$.

Note: By similiar analysis in Lemma 2, we have

$\lim_{\sigma\to 0} \frac{\mu_1}{a\sigma^{-d}} = 1$, $\lim_{\sigma\to 0} \frac{S_1}{\sigma^{-d}} = 0$

$\lim_{\sigma\to 0} \frac{\mu_2}{\tilde{a}\sigma^{-d}} = 1$, $\lim_{\sigma\to 0} \frac{S_2}{\sigma^{-d}} = 0$

$\lim_{\sigma\to 0} \frac{\mu_3}{\tilde{b}\sigma^{-d'}} = 1$, $\lim_{\sigma\to 0} \frac{\tilde{\mu}_3}{b\sigma^{-d'}} = 1$.

WLOG, assume $a \geq \tilde{a}$.

$$\begin{aligned}
R_0(c_0) = r \quad &\Rightarrow \quad P\left(\frac{f_{Z_1}(Z_1) f_{Z_2}(Z_2)}{f_{Z_1}(Z_1 + \frac{1}{\sigma}) f_{Z_2}(Z_2 + \frac{1}{\sigma})} \geq c_0\right) = r \\
&\Rightarrow \quad P(r_1 + r_2 \geq \log(c_0)) = r \\
&\Rightarrow \quad P\left(\frac{1}{S_2} \cdot \frac{r_1 - \mu_1}{S_1} + \frac{1}{S_1} \cdot \frac{r_2 - \mu_2}{S_2} \geq \frac{\log(c_0) - (\mu_1 + \mu_2)}{S_1 \cdot S_2}\right) = r \\
&\Rightarrow \quad P\left(\frac{\frac{1}{S_2} \cdot \frac{r_1 - \mu_1}{S_1} + \frac{1}{S_1} \cdot \frac{r_2 - \mu_2}{S_2}}{\sqrt{1/S_1^2 + 1/S_2^2}} \geq \frac{\log(c_0) - (\mu_1 + \mu_2)}{\sqrt{S_1^2 + S_2^2}}\right) = r \\
&\Rightarrow \quad \frac{\log(c_0) - (\mu_1 + \mu_2)}{\sqrt{S_1^2 + S_2^2}} = \alpha_r \\
&\Rightarrow \quad \log(c_0) = \mu_1 + \mu_2 + \alpha_r \sqrt{S_1^2 + S_2^2} \\
R_p(c_1, c_2) = r \quad &\Rightarrow \quad P\left(\frac{f_{Z_1}(Z_1)}{f_{Z_1}(Z_1 + \frac{1}{\sigma})} \geq c_1\right) P\left(\frac{f_{Z_2}(Z_2)}{f_{Z_2}(Z_2 + \frac{1}{\sigma})} \geq c_2\right) = r \\
&\Rightarrow \quad P(r_1 \geq \log(c_1)) P(r_2 \geq \log(c_2)) = r
\end{aligned}$$

Choose c_1, c_2 such that $P(r_1 \geq \log(c_1)) = P(r_2 \geq \log(c_2)) = \sqrt{r}$. So

$$\begin{aligned}
P(r_1 \geq \log(c_1)) = \sqrt{r} \quad &\Rightarrow \quad P\left(\frac{r_1 - \mu_1}{S_1} \geq \frac{\log(c_1) - \mu_1}{S_1}\right) = \sqrt{r} \\
&\Rightarrow \quad \frac{\log(c_1) - \mu_1}{S_1} = \beta_{\sqrt{r}} \\
&\Rightarrow \quad \log(c_1) = \beta_{\sqrt{r}} S_1 + \mu_1
\end{aligned}$$

Similiarly, $\log(c_2) = \tilde{\beta}_{\sqrt{r}} S_2 + \mu_2$

$$\begin{aligned}
S_0(c_0) &\geq \frac{p_1}{p_0+p_1+p_2} P\left(\frac{f_{Z_1}(Z_1) f_{Z_2}(Z_2-\frac{1}{\sigma})}{f_{Z_1}(Z_1+\frac{1}{\sigma}) f_{Z_2}(Z_2)} \geq c_0\right) \\
&= \frac{p_1}{p_0+p_1+p_2} P(r_1 - r_3 \geq \log(c_0)) \\
&= \frac{p_1}{p_0+p_1+p_2} P(r_1 - r_3 \geq \mu_1 + \mu_2 + \alpha_r \sqrt{S_1^2+S_2^2}) \\
S_p(c_1,c_2) &= \frac{p_0}{p_0+p_1+p_2} P(-\tilde{r}_3 \geq \log(c_1)) P(-r_3 \geq \log(c_2)) \\
&\quad + \frac{p_1}{p_0+p_1+p_2} P(r_1 \geq \log(c_1)) P(-r_3 \geq \log(c_2)) \\
&\quad + \frac{p_2}{p_0+p_1+p_2} P(-\tilde{r}_3 \geq \log(c_1)) P(r_2 \geq \log(c_2)) \\
&\leq \text{constant} \cdot (P(-r_3 \geq \tilde{\beta}_{\sqrt{r}} S_2 + \mu_2) + P(-\tilde{r}_3 \geq \beta_{\sqrt{r}} S_1 + \mu_1))
\end{aligned}$$

By Lemma 9, we get $\lim_{\sigma\to 0} \frac{P(-r_3 \geq \tilde{\beta}_{\sqrt{r}} S_2+\mu_2)}{P(Z_2 \geq \frac{1-\eta}{\sigma})} \leq 1$ and $\lim_{\sigma\to 0} \frac{P(-\tilde{r}_3 \geq \beta_{\sqrt{r}} S_1+\mu_1)}{P(Z_1 \geq \frac{1-\xi}{\sigma})} \leq 1$ for $\forall \eta, \xi > 0$. We state and derive Lemma 9' below for later use.

Lemma 9': $\lim_{\sigma\to 0} \frac{P(-r_3 \geq \tilde{\beta}_{\sqrt{r}} S_2+\mu_2) + P(-\tilde{r}_3 \geq \beta_{\sqrt{r}} S_1+\mu_1)}{2\max(P(Z_1 \geq \frac{1-\eta}{\sigma}), P(Z_2 \geq \frac{1-\eta}{\sigma}))} \leq 1$ for $\forall 0 < \eta < 1$.

Proof:

$$\begin{aligned}
& \lim_{\sigma\to 0} \frac{P(-r_3 \geq \tilde{\beta}_{\sqrt{r}} S_2+\mu_2) + P(-\tilde{r}_3 \geq \beta_{\sqrt{r}} S_1+\mu_1)}{2\max(P(Z_1 \geq \frac{1-\eta}{\sigma}), P(Z_2 \geq \frac{1-\eta}{\sigma}))} \\
= & \lim_{\sigma\to 0} \frac{P(-r_3 \geq \tilde{\beta}_{\sqrt{r}} S_2+\mu_2)}{2\max(P(Z_1 \geq \frac{1-\eta}{\sigma}), P(Z_2 \geq \frac{1-\eta}{\sigma}))} + \lim_{\sigma\to 0} \frac{P(-\tilde{r}_3 \geq \beta_{\sqrt{r}} S_1+\mu_1)}{2\max(P(Z_1 \geq \frac{1-\eta}{\sigma}), P(Z_2 \geq \frac{1-\eta}{\sigma}))} \\
\leq & \frac{1}{2} + \frac{1}{2} \\
= & 1
\end{aligned}$$

Also, we need to do some minor changes to Lemma 8, 7, 6.

Lemma 8': $\lim_{\sigma\to 0} \frac{P(r_1 - r_3 \geq \mu_1+\mu_2+\alpha_r\sqrt{S_1^2+S_2^2})}{P(Z_1 \geq \frac{u_\delta}{\sigma}, Z_2 \geq \frac{v_\delta}{\sigma})} \geq 1$, for some $u_\delta > 0, v_\delta > 0$ such that

$u_\delta^d + \frac{\tilde{a}}{a} v_\delta^d < \frac{\tilde{a}}{a}$.

Lemma 7': There exists $\epsilon_M > 0$, such that $\forall \epsilon < \epsilon_M, h_\epsilon(u,v) \doteq (1+u)^d - \frac{a+\epsilon}{a-\epsilon} u^d + \frac{\tilde{a}-\epsilon}{a-\epsilon} v^d - \frac{\tilde{b}+\epsilon}{a-\epsilon}(1-v)^d$ is increasing function of u, v, where $u_0 \le u \le 1, v_0 \le v \le 1$ for $\forall 0 < u_0, v_0 < 1$.

Lemma 6': Equation $-u^d + (1+u)^d + \frac{\tilde{a}}{a} v^d - \frac{\tilde{b}}{a}(1-v)^d = 1 + \frac{\tilde{a}}{a}$ has a solution u^*, v^* such that $u^* > 0, v^* > 0$ and $(u^*)^d + \frac{\tilde{a}}{a}(v^*)^d < \frac{\tilde{a}}{a}$.

It is simple to check that all the new Lemmas 6',7',8' and Lemma 10 hold. In other words, we just generalize the notation a bit, and apply the same analysis. The related generalized notations are listed below.

$$\begin{aligned}
g_1(Z_1, Z_2) &= -\phi(Z_1)|Z_1|^d + \phi(Z_1 + \frac{1}{\sigma})|Z_1 + \frac{1}{\sigma}|^d + \psi(Z_2)|Z_2|^d - \psi(Z_2 - \frac{1}{\sigma})|Z_2 - \frac{1}{\sigma}|^{d'} \\
h(u,v) &= -u^d + (1+u)^d + \frac{\tilde{a}}{a} v^d - \frac{\tilde{b}(1-v)^{d'}}{a} \sigma^{d-d'} \\
h_\epsilon(u,v) &= (1+u)^d - \frac{a+\epsilon}{a-\epsilon} u^d + \frac{\tilde{a}-\epsilon}{a-\epsilon} v^d - \frac{\tilde{b}+\epsilon}{a-\epsilon}(1-v)^{d'} \sigma^{d-d'} \\
F(u,v) &= h(u,v) - 1 - \frac{\tilde{a}}{a}
\end{aligned}$$

Combine Lemma 8', Lemma 9' and Lemma 10, and fix a sufficent small η,

$$\begin{aligned}
\lim_{\sigma \to 0} \frac{S_0(c_0)}{S_p(c_1, c_2)} &\ge \text{constant} \cdot \lim_{\sigma \to 0} \frac{P(Z_1 \ge \frac{u_\delta}{\sigma}, Z_2 \ge \frac{v_\delta}{\sigma})}{\max(P(Z_1 \ge \frac{1-\eta}{\sigma}), P(Z_2 \ge \frac{1-\eta}{\sigma}))} \\
&= \text{constant} \cdot \lim_{\sigma \to 0} \frac{P(Z_1 \ge \frac{u_\delta}{\sigma}) P(Z_2 \ge \frac{v_\delta}{\sigma})}{\max(P(Z_1 \ge \frac{1-\eta}{\sigma}), P(Z_2 \ge \frac{1-\eta}{\sigma}))} \\
&= \text{constant} \cdot \lim_{\sigma \to 0} \frac{e^{-a(\frac{u_\delta}{\sigma})^d} e^{-\tilde{a}(\frac{v_\delta}{\sigma})^d + o(\frac{1}{\sigma^d})}}{e^{-\tilde{a}(\frac{1-\eta}{\sigma})^d + o(\frac{1}{\sigma^d})}} \\
&\ge \text{constant} \cdot \lim_{\sigma \to 0} e^{\frac{a}{\sigma^d}(\frac{\tilde{a}}{a}(1-\eta)^d - u_\delta^d - \frac{\tilde{a}}{a} v_\delta^d) + o(\frac{1}{\sigma^d})} \\
&= \infty
\end{aligned}$$

Lemma 12: *Consider i.i.d noise with double exponential distribution, i.e., $f(z) = \frac{1}{2}e^{-|z|}$, the ratio is finite.*

Proof: Following the same notation, in this case, r_2 and r_3 have the same distribution. The argument goes as follows.

Because Z_2 and $-Z_2$ share the same distribution by the symmetricity of $f(z)$, this implies $\log\frac{f(-Z_2)}{f(-Z_2+\frac{1}{\sigma})}$ and $\log\frac{f(Z_2)}{f(Z_2+\frac{1}{\sigma})}$ share the same distribution, and we know $\log\frac{f(-Z_2)}{f(-Z_2+\frac{1}{\sigma})}$ $= \log\frac{f(Z_2)}{f(Z_2-\frac{1}{\sigma})}$ by symmetricity of $f(z)$, so $\log\frac{f(Z_2)}{f(Z_2+\frac{1}{\sigma})}$ and $\log\frac{f(Z_2)}{f(Z_2-\frac{1}{\sigma})}$ share the same distribution, i.e., r_2 and r_3 have the same distribution.

For $i = 1, 2$

$$\begin{aligned} r_i &= |Z_i + \frac{1}{\sigma}| - |Z_i| \\ \mu_i &= E(r_i) \\ S_i &= s.d(r_i) \end{aligned}$$

Note: $\frac{1}{\sigma} - |Z_1| - |Z_1| \leq r_1 \leq |Z_1| + \frac{1}{\sigma} - |Z_1| \Rightarrow \frac{1}{\sigma} - 2E|Z_1| \leq \mu_1 \leq \frac{1}{\sigma}$.

Note: $var(r_1) \leq 2(var|Z_1 + \frac{1}{\sigma}| + var|Z_1|) \leq 2(var(Z_1 + \frac{1}{\sigma}) + var|Z_1|) = 2(var(Z_1) + var|Z_1|)$.

Let $A \doteq -\alpha_r\sqrt{2(var(Z_1) + var|Z_1|)} + 4E|Z_1|$. (Assume $\alpha_r < 0, \beta_{\sqrt{r}} < 0$, true for big r.)

c_0 and c_1 from $R_0(c_0) = R_p(c_1) = r$ are computed, they are in the same form as before.

So we have $\lim_{\sigma\to 0}\frac{S_0(c_0)}{S_p(c_1)} = \lim_{\sigma\to 0}$ constant $\cdot \frac{P(|Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq 2\mu_1+\sqrt{2}\alpha_r S_1)}{P(|Z_2|-|Z_2+\frac{1}{\sigma}|\geq \mu_1+\beta_{\sqrt{r}}S_1)}$

Consider the numerator, $|Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq 2\mu_1+\sqrt{2}\alpha_r S_1$

$$\begin{aligned} \Rightarrow \quad & |Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq \frac{2}{\sigma}-4E|Z_1|+\sqrt{2}\alpha_r\sqrt{2(var(Z_1)+var|Z_1|)} \\ \Leftrightarrow \quad & |Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq \frac{2}{\sigma}-A \end{aligned}$$

Since $|Z_1+\frac{1}{\sigma}|-|Z_1|\leq\frac{1}{\sigma}$ and $|Z_2|-|Z_2+\frac{1}{\sigma}|\leq\frac{1}{\sigma}$,
then $\{|Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq\frac{2}{\sigma}-A\}\subseteq\{|Z_1+\frac{1}{\sigma}|-|Z_1|\geq\frac{1}{\sigma}-A$ & $|Z_2|-|Z_2+\frac{1}{\sigma}|\geq\frac{1}{\sigma}-A\}$.

We know $\{|Z_1+\frac{1}{\sigma}|-|Z_1|\geq\frac{1}{\sigma}-A\}\subseteq\{Z_1\geq-\frac{A}{2}\}$ and $\{|Z_2|-|Z_2+\frac{1}{\sigma}|\geq\frac{1}{\sigma}-A\}\subseteq\{Z_2\leq-\frac{1}{\sigma}+\frac{A}{2}\}$.

Finally, we have $\{|Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq 2\mu_1+\sqrt{2}\alpha_r S\}\subseteq\{Z_1\geq-\frac{A}{2}$ & $Z_2\leq-\frac{1}{\sigma}+\frac{A}{2}\}$. Equivalently $P(Z_1\geq-\frac{A}{2},Z_2\leq-\frac{1}{\sigma}+\frac{A}{2})\geq P(r_1-r_2\geq 2\mu_1+\sqrt{2}\alpha_r S)$.

Consider the denominator, since $\{Z_2\leq-\frac{1}{\sigma}\}\subseteq\{|Z_2|-|Z_2+\frac{1}{\sigma}|=\frac{1}{\sigma}\geq\mu_1+\beta_{\sqrt{r}}S_1\}$, therefore $P(|Z_2|-|Z_2+\frac{1}{\sigma}|\geq\mu_1+\beta_{\sqrt{r}}S_1\})\geq P(Z_2\leq-\frac{1}{\sigma})$.

Combine both inequalities,

$$\begin{aligned} \lim_{\sigma\to 0}\frac{S_0(c_0)}{S_p(c_1)} &= \text{constant}\cdot\lim_{\sigma\to 0}\frac{P(|Z_1+\frac{1}{\sigma}|-|Z_1|+|Z_2|-|Z_2+\frac{1}{\sigma}|\geq 2\mu_1+\sqrt{2}\alpha_r S_1)}{P(|Z_2|-|Z_2+\frac{1}{\sigma}|\geq\mu_1+\beta_{\sqrt{r}}S_1)} \\ &\leq \text{constant}\cdot\lim_{\sigma\to 0}\frac{P(Z_1\geq-\frac{A}{2})P(Z_2\leq-\frac{1}{\sigma}+\frac{A}{2})}{P(Z_2\leq-\frac{1}{\sigma})} \\ &= \text{finite constant} \end{aligned}$$

Chapter 4

Computational Framework

In this chapter, we enumerate the details of a manually designed hierarchical structure for license-reading application. Then, based on such a design, we describe the dependency structure among all the introduced bricks to gain a better understanding of the complexity of the scene annotation problem. Next, we elaborate on the building blocks of a compositional machine, including concepts, algorithms and implementation details. Finally, we demostrate different themes by running the constructed compositional machine under different settings and conclude with a comment on the current model.

4.1 Hierarchical Structure of the License Plate

In Figure 2.2, we see that bricks are arranged in layers. For the first four layers, string-related bricks and boundary-related bricks are mixed together. But since typical string and license boundary don't share subparts, hierarchical structures for string and boundary are constructed and discussed separately. Then they are combined to form a license plate. Pictures of all part filters and context-free rules on how they are composed into objects are listed in the Appendix.

4.1.1 Hierarchical Structure for String

$x_s^{l,k,(i,j)}$ represents the state of a brick, which is at (i,j) entry of k^{th} sublattice on the l^{th} layer of the string hierarchy. Bricks in the k^{th} sublattice have label (or type) k. Equivalently it is represented as x_s^{β} where $\beta = (l, k, (i,j))$. Subscript s can be omitted if there is no confusion.

4.1.1.1 First/Terminal Layer (character part layer)

The state configuration for this layer is $x_s^{1,k,(i,j)}$, where $k \in \{1..43\}$ represent 43 distinct labels (i.e., part filters of characters). Each brick covers a non-overlapping 5×5 image region (which is defined as its receptive field, abbreviated by RF, for details see notes

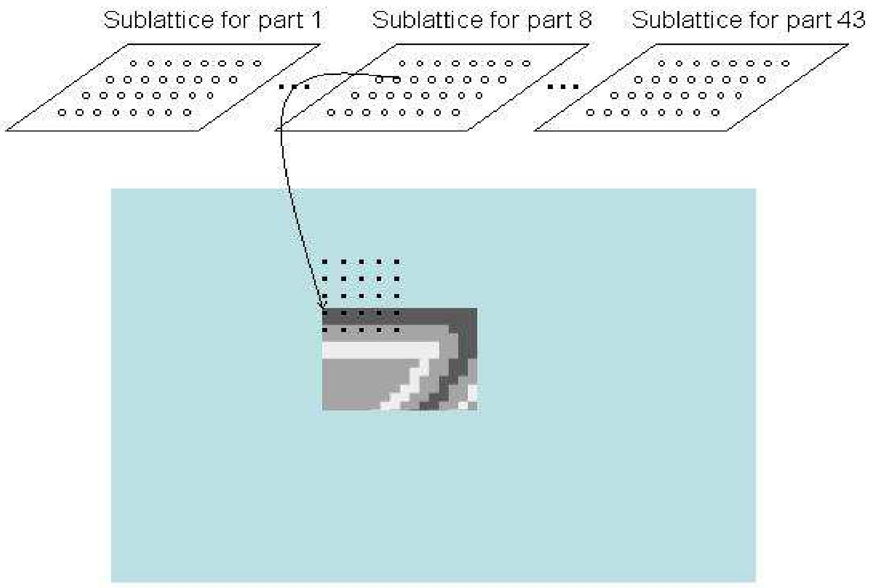


Figure 4.1: At the terminal layer (i.e., layer = 1), in $8th$ sublattice (corresponding to the upper part of character "3"), the (2,2) brick points to $4th$ out of 25 possible locations. That is, $x_s^{1,8,(2,2)} = 4$. The big blue rectangle refers to the test image.

at §4.1.4.4) with its state $x \in \{0, 1, ..25\}$, meaning the upper-left corner of that filter patch can be either nowhere or anywhere in the region. For example, refer to Figure 4.1 and its description. Details about matching the part filter patch with the test image will be covered in the data model (§4.3).

To estimate the state probabilities of each brick at this layer, suppose we have estimated the probability for each filter patch when it's present or absent on a pixel basis (for details, see §4.3.1.2) and assume translation invariance, then we have $\epsilon_0^\beta = (\epsilon_0)^{25}$ and $\epsilon_i^\beta = \frac{1-(\epsilon_0)^{25}}{25}$ for i=1..25, where ϵ_0 is the absence probability for each pixel and β is a brick at this layer. Note: ϵ_0 is label dependent.

No attribute is involved at this layer.

Consider a composition machine with only one layer (i.e., terminal layer); the likelihood ratio of the posterior distributions of a particular interpretation I (i.e., state configuration of the terminal layer X^τ in I) versus the null interpretation (i.e., all the bricks are off), given image Y, can be written as

$$\begin{aligned}\frac{P(X^\tau(I)|Y)}{P(X(\phi)|Y)} &= \frac{P(Y|X^\tau(I))}{P(Y|X(\phi))}\cdot\frac{P(X^\tau(I))}{P(X(\phi))}\\ &= \frac{P(R|X^\tau(I))}{P(R|X(\phi))}\cdot\prod_{\beta\in\mathcal{B}_1}\frac{\epsilon^\beta_{x^\beta}}{\epsilon^\beta_0}\end{aligned}$$

where R is the rank sum statistics of image Y under interpretation I, which is exponentially distributed, conditioned on $X^\tau(I)$, and approximately normally distributed conditioned on $X(\phi)$ (for details, see §4.3.1.1), and $\mathcal{B}_1$ is the set of all bricks at this layer. The second equality holds by the assumptions in §2.2, that is, R is sufficient statistics for data referenced by active terminal bricks; and data which are not referenced by any terminal brick are uniformly independently distributed, implying that R is also sufficient. The involved combinatorics constants in the numerator and denominator are cancelled out by the assumptions. For more details, refer to [18]. We will use this likelihood ratio during the computational scheme later to decide whether to activate a terminal brick or not.

4.1.1.2 Second Layer (specific digit/letter layer)

The state configuration for this layer is $x_s^{2,k,(i,j)}$, where $k \in \{1..36\}$ represent 36 characters (i.e., digit 0..9 and letter A..Z). Each character consists of 2 or 3 parts. Each sublattice (corresponding to each character) at this layer gets coarser by a factor of 5×4 compared to the sublattice (corresponding to its upper part) at the previous layer, meaning each character brick can be instantiated with its upper-part being anywhere in the 5×4 grid of the sublattice (corresponding to the upper part) at the terminal layer (or with its upper-left corner, that is, the upper-left corner of its upper part, being anywhere in the 25×20 image region, which is defined as its RF). Fix a character's upper-part brick in one of the 5×4 grids, its lower part(s) brick(s) can float

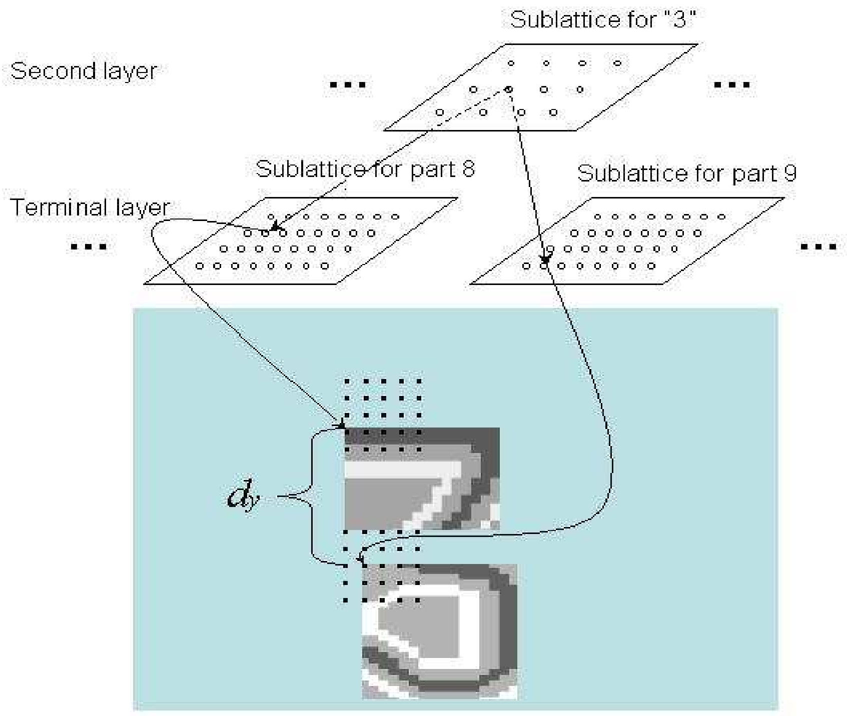


Figure 4.2: A particular instantiation of character "3" is shown. In the sublattice for "3" at the second layer, the (2,2) brick selects (is instantiated by) a subset of children bricks (designated by its state) from a collection of allowed children sets. In this figure, the chosen children are brick (2,2) in 8th sublattice (corresponding to the upper part of "3") and brick (4,2) in 9th sublattice (corresponding to lower part of "3") at the terminal layer; and $x_s^{1,8,(2,2)} = 4$, $x_s^{1,9,(4,2)} = 8$.

within a neighborhood. For a two-part character, the fluctuation of the lower part, given a fixed upper part, is within a neighborhood of grid size 3×3 in its sublattice. So the total number of states of a two-part character brick is $5 \times 4 \times 3 \times 3 + 1 = 181$ (Note: $+1$ comes from the null state). For a three-part character, with its upper part fixed, we only allow 20 reasonable spatial configurations for the lower two parts, instead of all possible $9 \times 9 = 81$ configurations. So the total number of states is $5 \times 4 \times 20 + 1 = 401$ for a three-part character brick. An example of a particular instantiation of a second layer brick is shown in Figure 4.2.

To estimate the state probabilities, we fix a character, first count its occurrence among the training set and divide the count by the number of training images; its average number per image is then divided by the total number of its bricks, then further divided by the number of its children sets. This is the estimated probability of each active state for each brick of this character. The null probability for such brick is one minus the summation of its active state probabilities.

Involved attributes at this layer and their (composed and null) distributions are described below.

For a 2-part character (e.g, "5" in Figure 2.3): attribute is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance (details see notes §4.1.4.2) along horizontal direction (x-distance) and vertical direction (y-distance) respectively between the upper-left corners of the two parts in an instantiation of a character brick. For example, in Figure 4.2, the marked d_y measures the y-distance attribute in that particular instantiation of "3"; $d_x = 1$ in this case, because the two 5×5 grids in the image are vertically aligned. Under the composition, we assign a product of two *truncated* normal distributions (for details, see notes §4.1.4.5) over the attribute (i.e., x-distance and y-distance). Under the null, we assign a product of two triangular distributions over the attribute, which is achieved through enumerating all the possible signed distances between the two corners. For a detailed justification, see §4.3.2.2.

For a 3-part character (e.g, "0" in Figure 2.3): attribute is the signed pairwise distance between the upper-left corners of adjacent parts. There are two pairs involved, i.e., the upper-part and the middle-part, the middle-part and the lower-part. Under the composition, we assign a product of four truncated normal distributions over the attribute (i.e., two pairwise x-distances and two pairwise y-distances). Under the null, we assign a product of four triangular distributions over the same attribute.

Consider a composition machine with only two layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the second layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta\in\mathcal{B}_2} \frac{\epsilon_i^\beta/\epsilon_0^\beta}{\prod_{\alpha\in C_i^\beta - B(I)}(1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the second layer and at the second layer I' includes the compositions that lead to each active β at state i, while I does not), $\mathcal{B}_2$ is the set of all bricks at this layer, C_i^β is the *ith* children set of β and a_i^β is the context-sensitive attribute related with brick β at state i. Again, we will use this likelihood ratio in the computational scheme later to decide whether to activate a second layer brick.

4.1.1.3 Third Layer (general digit/letter layer)

The state configuration for this layer is $x_s^{3,k,(i,j)}$, where $k \in \{1,2\}$ represent digit and letter respectively. The two sublattices keep the same resolution as the sublattice at the second layer (i.e., not getting coarser). The state of each digit brick $x \in \{0,1,..10\}$ (10 children sets), meaning no digit or digit-0 brick, digit-1 brick,.., digit-9 brick, is instantiated on the second layer at the corresponding position respectively. Similarly the state of each letter brick $x \in \{0,1,..26\}$ (26 children sets), meaning no letter or letter-A brick, .. letter-Z brick, is instantiated on the second layer at the corresponding position respectively. Note: a general character brick on the third layer has the same RF as a specific character brick on the second layer.

To estimate the state probabilities, we fix a digit (letter) brick, apply the similar procedure (counting and dividing) as for a specific character brick to estimate its state probability vector.

No attribute is involved at this layer.

Consider a composition machine with only three layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the third layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta\in\mathcal{B}_3} \frac{\epsilon_i^\beta/\epsilon_0^\beta}{\prod_{\alpha\in C_i^\beta - B(I)}(1-\epsilon_0^\alpha)}$, where I', I are two close interpretations (they have the same state configurations before the third layer and at the third layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_3$ is the set of all bricks at this layer.

4.1.1.4 Fourth Layer (partial-string layer)

The state configuration for this layer is $x_s^{4,k,(i,j)}$, where $k \in \{1..4\}$ represent 2-letter, 3-digit, 3-letter, 4-digit respectively. Each sublattice (corresponding to each type of partial string) at this layer gets coarser by a factor of 2×3 compared to the sublattice (corresponding to its leftmost character) at the previous layer, meaning each partial string can be instantiated with its leftmost character being anywhere in the 2×3 grid at the third layer (or with its upper-left corner, that is, the upper-left corner of its leftmost character, being anywhere in the 50×60 image region, which is defined as its RF). Fix the leftmost character of a partial-string in one of the 2×3 grid, the following character bricks can float within a neighborhood. For a 2-letter partial string, with its left letter fixed in one of the 2×3 grids, the right letter can fluctuate within a neighborhood of grid size 3×2 in its sublattice. So the total number of states of a two-letter brick is $2 \times 3 \times 3 \times 2 + 1 = 37$. For a 3-character partial string, with its leftmost character fixed, we only allow 20 reasonable spatial configurations for the right two characters instead of all possible $6 \times 6 = 36$ configurations. The total number of states is $2 \times 3 \times 20 + 1 = 121$ for a 3-character brick. For a 4-digit partial string, fix its leftmost digit, we only allow 56 reasonable spatial configurations for the right three digits instead of all possible $6 \times 6 \times 6 = 216$ configurations. The total number of states is $2 \times 3 \times 56 + 1 = 337$ for a 4-digit brick.

To estimate the state probabilities, we fix a type of partial string, apply the similar procedure (counting and dividing) to estimate its state probability vector. In this case, we have to be careful with the counting process. For example, how many 3-digit partial strings are counted in a given 4-digit partial string? The counts should be consistent with the ways in which three digits can be composed into a 3-digit partial string (i.e., 3-digit brick's children set). For example, one 4-digit string will be counted as 3-digit four times in our setting, because each of them could be instantiated as a 3-digit partial string.

Involved attributes at this layer and their distributions are described below.

For 2-letter: attribute is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively between upper-left corners of the two letters. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, similar as in a 2-part character, we assign a product of two triangular distributions over the attribute.

For 3-digit/letter: attribute is signed pairwise distances (along x-direction and y-direction) between upper-left corners of adjacent characters (two pairs are involved, i.e., the leftmost character and the middle one; the middle one and the rightmost one). Under the composition, we assign a product of four truncated normal distributions over the attribute (i.e., the two pairwise x-distances and two pairwise y-distances). Under the null, similar as in a 3-part character, we assign a product of four triangular distributions over the same attribute.

For 4-digit: attribute is signed pairwise distances (along x-direction and y-direction) between upper-left corners of adjacent digits (three pairs are involved, i.e., the leftmost digit and the next-to-leftmost, the next-to-leftmost and the next-to-rightmost, the next-to-rightmost and the rightmost). Under the composition, we assign a product of six truncated normal distributions over the attribute (i.e., three pairwise x-distances and three pairwise y-distances). Under the null, we assign a product of six triangular distributions over the same attribute.

Consider a composition machine with only four layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the fourth layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta \in \mathcal{B}_4} \frac{\epsilon_i^\beta / \epsilon_0^\beta}{\prod_{\alpha \in C_i^\beta - B(I)} (1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the fourth layer and at the fourth layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_4$ is the set of all bricks at this layer.

4.1.1.5 Fifth Layer (string layer)

The state configuration for this layer is $x_s^{5,1,(i,j)}$. The lattice of string bricks at this layer gets coarser by a factor of 2×2 compared to the sublattice (corresponding to its left partial string) at previous layer, meaning each string can be instantiated with its left partial string being anywhere in the 2×2 grids at the fourth layer (or with its upper-left corner, that is, the upper-left corner of its left partial string, being anywhere in the 100×120 image region, which is defined as its RF). Each string consists of two partial-strings, either 4-digit plus 2-letter or 3-digit plus 3-letter. Fix the left partial-string of a string in one of the 2×2 grid, the right partial-string brick can float within a neighborhood of grid size 3×4 in its sublattice. The total number of states for a string brick is $2 \times 2 \times 3 \times 4 + 1 = 49$.

The method used to estimate the state probabilities is similar as before (i.e., counting and dividing).

The attributes for a string is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively between the upper-left corners of the two composing partial strings. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, we assign a product of two triangular distributions over the attribute.

Consider a composition machine with five layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the fifth layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta \in \mathcal{B}_5} \frac{\epsilon_i^\beta / \epsilon_0^\beta}{\prod_{\alpha \in C_i^\beta - B(I)} (1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the fifth layer and at the fifth layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_5$ is the set of all bricks at this layer.

4.1.2 Hierarchical Structure for Plate Boundary

$x_b^{l,k,(i,j)}$ represents the state of a brick, which is at (i,j) entry of k^{th} sublattice on the l^{th} layer of the boundary (frame) hierarchy. Bricks in the k^{th} sublattice have label (or type) k. Equivalently this is represented as x_b^{β} where $\beta = (l,k,(i,j))$. Subscript b can be omitted if there is no confusion.

4.1.2.1 First/Terminal Layer (medium line layer)

The state configuration for this layer is $x_b^{1,k,(i,j)}$, where $k \in \{1..4\}$ represent four distinct labels: medium vertical line, medium horizontal line and their flipped version (see Figures 4.3-4.6). Each brick covers a 5×5 image region (which is defined as its RF) with its state $x \in \{0, 1, ..25\}$, meaning the upper-left corner (of the two medium lines) or the lower-right corner (of their flipped version) can be either nowhere or anywhere in the region.

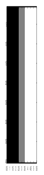


Figure 4.3: medium vertical line

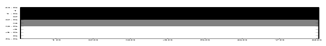


Figure 4.4: medium horizontal line

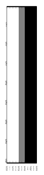


Figure 4.5: flipped medium vertical line

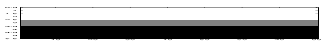


Figure 4.6: flipped medium horizontal line

The procedure used to estimate the state probabilities of each brick at this layer is the same as for the character part, i.e., $\epsilon_0^{\beta} = (\epsilon_0)^{25}$ and $\epsilon_i^{\beta} = \frac{1-(\epsilon_0)^{25}}{25}$ for i=1..25, where ϵ_0 is the absence probability for each pixel and β is a brick at this layer. Note: ϵ_0 is label dependent.

No attribute is involved at this layer.

Consider a composition machine with only one layer (i.e., terminal layer), the likelihood ratio of the posterior distributions of a particular interpretation I (i.e., state configuration of the terminal layer X^{τ} in I) versus the null interpretation, given image Y, can be written as

$$\begin{aligned} \frac{P(X^{\tau}(I)|Y)}{P(X(\phi)|Y)} &= \frac{P(Y|X^{\tau}(I))}{P(Y|X(\phi))} \cdot \frac{P(X^{\tau}(I))}{P(X(\phi))} \\ &= \frac{P(R|X^{\tau}(I))}{P(R|X(\phi))} \cdot \prod_{\beta \in \mathcal{B}_1} \frac{\epsilon^{\beta}_{x^{\beta}}}{\epsilon^{\beta}_{0}} \end{aligned}$$

where R is the rank sum statistics of image Y under interpretation I, which is exponentially distributed conditioned on $X^{\tau}(I)$ and approximately normally distributed conditioned on $X(\phi)$ (details see §4.3.1.1), and $\mathcal{B}_1$ is the set of all bricks at this layer. The second equality holds by the independence assumption and the sufficiency assumption in §2.2.

4.1.2.2 Second Layer (long line layer)

The state configuration for this layer is $x_b^{2,k,(i,j)}$, where $k \in \{1..4\}$ represent long vertical line, long horizontal line and their flipped version. They look the same as their corresponding medium lines except the length is doubled, that is, each long line consists of 2 medium lines of the same type. The first part of a long vertical line is defined as its upper vertical medium line; the first part of a long horizontal line is defined as its left medium horizontal line; the first part of a flipped long vertical line is defined as its lower vertical medium line; the first part of a flipped long horizontal line is defined as its right medium horizontal line. Each sublattice (corresponding to each long line) at this layer gets coarser by a factor of 5×4 compared to the sublattice (corresponding to its first part) at the previous layer, meaning each long line can be instantiated with its first part being anywhere in the 5×4 grid at the terminal layer (or equivalently, with its upper-left corner (if labelled 1, 2) or lower-right corner (if labelled 3,4), that is, the upper-left corner of its first part or the lower-right corner of its first part, being anywhere in the 25×20 image region, which is defined as its RF). Fix a long line's first part in one of the 5×4 grids, the second part brick can float within a neighborhood of grid size 3×3 in its sublattice. So the total number of states for each long-line brick is $5 \times 4 \times 3 \times 3 + 1 = 181$.

The state probabilities are estimated by empirical counting as before.

The attribute for a long line brick of each type is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively, between the same corners of two medium lines of the same type. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, we assign a product of two triangular distributions over the same attribute.

Consider a composition machine with only two layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the second layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta \in \mathcal{B}_2} \frac{\epsilon_i^\beta / \epsilon_0^\beta}{\prod_{\alpha \in C_i^\beta - B(I)} (1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the second layer and at the second layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_2$ is the set of all bricks at this layer.

4.1.2.3 Third Layer (L-shape layer)

The state configuration for this layer is $x_b^{3,k,(i,j)}$, where $k \in \{1..2\}$ represent upper L shape (i.e., the upper-left half of the boundary, composed of a long vertical line and a long horizontal line) and lower L shape (i.e., the lower-right half of the boundary, composed of the flipped long vertical and horizontal lines). Both sublattices (corresponding to the two L shapes) at this layer get coarser by a factor of 2×3 compared to the sublattices (corresponding to their vertical parts) at the previous layer, meaning each L-shape brick can be instantiated with its vertical part being anywhere in the 2×3 grid at the second layer (or with its corner, that is, the upper-left corner of its vertical part for upper L shape and the lower-right corner of its vertical part for lower

L shape, being anywhere in the 50×60 image region, which is defined as its RF). Fix L-shape's vertical part in one of the 2×3 grids, the horizontal part can float within a neighborhood of grid size 3×3 in its sublattice. So the total number of states for each L-shape brick is $2 \times 3 \times 3 \times 3 + 1 = 55$.

The state probabilities are estimated in the same manner as before.

The attribute for an L-shape brick of each type is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively, between the close ends of its horizontal part and vertical part. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, we assign a product of two triangular distributions over the same attribute.

Consider a composition machine with only three layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the third layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta \in \mathcal{B}_3} \frac{\epsilon_i^\beta / \epsilon_0^\beta}{\prod_{\alpha \in C_i^\beta - B(I)} (1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the third layer and at the third layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_3$ is the set of all bricks at this layer.

4.1.2.4 Fourth Layer (plate boundary layer)

The state configuration for this layer is $x_b^{4,1,(i,j)}$. Each boundary consists of an upper L-shape and a lower L-shape. The lattice of boundary bricks at this layer gets coarser by a factor of 2×2 compared to the sublattice (corresponding to its upper L-shape) at the previous layer, meaning each boundary brick can be instantiated with its upper L-shape brick being anywhere in the corresponding 2×2 grid at the third layer (or with its upper-left corner, that is, its upper L shape's corner, being anywhere in the 100×120 image region, which is defined as its RF). Fix a boundary's upper L-shape brick in one of the 2×2 grids, the lower L-shape brick can float within a neighborhood of grid size 3×3 in its sublattice. The number of states for each boundary brick is $2 \times 2 \times 3 \times 3 + 1 = 37$.

The state probabilities are estimated in the same manner as before.

The attribute for each boundary brick is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively, between the corners of its upper L-shape and lower L-shape. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, we assign a product of two triangular distributions over the same attribute.

Consider a composition machine with four layers. By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the fourth layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta\in\mathcal{B}_4} \frac{\epsilon_i^\beta/\epsilon_0^\beta}{\prod_{\alpha\in C_i^\beta - B(I)}(1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have the same state configurations before the fourth layer and at the fourth layer I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_4$ is the set of all bricks at this layer.

4.1.3 License Plate

Finally, we combine disjoint string hierarchical structure and boundary hierarchical structure together to form license plate objects at our top layer (the sixth layer).

According to the way we construct the two hierarchical structures, we get two of the same size 5×5 lattices of string bricks and boundary bricks. The lattice for license plate doesn't get coarser. Fix a license plate's boundary at a position on the fourth layer (in the hierarchical structure for boundary), the string brick can float within a neighborhood of grid size 2×2 in its lattice (on the fifth layer in the hierarchical structure for string). The number of states for each license brick is $2\times 2+1 = 5$. We only have $5 \times 3 = 15$ license plate bricks (instead of $5 \times 5 = 25$) in total at this layer, because the right two column of license bricks will only refer to truncated license plates.

The state probabilities are estimated in the same way as before.

The attribute for each license brick is a two-vector (d_x, d_y), where d_x, d_y measure the signed distance along x-direction and y-direction respectively, between the upper-left corners of boundary and string. Under the composition, we assign a product of two truncated normal distributions over the attribute. Under the null, we assign a product of two triangular distributions over the same attribute.

Consider a composition machine with six layers (the first four layers are mixed with both string-related and boundary-related bricks and the fifth layer contains string bricks). By simple calculation using Eqn 2.2, the likelihood ratio of composition versus non-composition at the six layer, given image Y, can be written as $\frac{P(I'|Y)}{P(I|Y)} = \prod_{\beta \in \mathcal{B}_6} \frac{\epsilon_i^\beta / \epsilon_0^\beta}{\prod_{\alpha \in C_i^\beta - B(I)} (1-\epsilon_0^\alpha)} \times \frac{p^c_{a_i^\beta}(a_i^\beta)}{p^0_{a_i^\beta}(a_i^\beta)}$, where I', I are two close interpretations (they have same state configurations up to the fifth layer, including the string layer and boundary layer, but at the license plate layer (sixth layer) I' includes the compositions that lead to each active β at state i, while I does not) and $\mathcal{B}_6$ is the set of all bricks at this layer.

4.1.4 Remarks

4.1.4.1 Notes about the Structure

Depending on the computational scheme, sometimes only string hierarchy is visited, sometimes both hierarchies (string and boundary) are visited in a given order, and sometimes they are visited simultaneously. When they are visited at the same time, both the string-related and boundary-related bricks need to be considered; that is, the attributes involved for each layer are the union of attributes for string-related and boundary-related bricks and the involved likelihood ratio is the joint likelihood ratio. Note: RF of bricks on the same layer may have a different size and it only depends on a brick's type.

4.1.4.2 Notes about Signed Distance

Signed distance comes from the fact that we explicitly put an order on the children bricks under composition when computing the attribute. For example, given that a character is composed by 2/3 parts in a vertical way, we subtract the (x,y) coordinate of the upper-left corner of the upper part from the (x,y) coordinate of the upper-left corner of the adjacent lower part to get the attribute (relative distance) between them. Say, digit “0” consists of an upper arc, middle parallel lines, and a lower arc. If we know the height of “0” is more or less fixed at 40 pixels, three involved parts can be designed all at a height around 15 pixels with some overlap between adjacent parts along the vertical direction to form a “0”. In this case, we assume the horizontal distances between neighboring parts is around 0. So we assign a product of two truncated normal distributions with mean 0 pixels and mean 13 pixels over the x-distance and y-distance (encouraging 2 pixels overlapping in y-direction) between every two adjacent parts. Proper variances are also set to allow a little pertubation. For a partial string which is composed by 2/3/4 characters in a horizontal way, we define its attributes in a similar fashion. In a word, composition could take place mainly along vertical direction, say 3-part character; or mainly along horizontal direction, say partial string; or along both directions, say license plate; or along any direction, say L-shape.

4.1.4.3 Notes about Bits Gain

For each layer, we compute the likelihood ratio of composition versus non-composition as mentioned above. Consider an interpretation I, which is formed layer by layer through building up the composition machine, the ratio of the posterior probabilities over the final I versus null interpretation given the test image Y is the product of each layer's likelihood ratio (of composition versus non-composition) in I by a telescoping argument. The log likelihood ratio of I with respect to null interpretation (i.e., $\log \frac{P(Y,I)}{P(Y,\phi)}$) is equivalent to the bits difference of their shannon codes under the joint distribution. More precisely, it is the bits gain of I with respect to the null. We calculate it by summing up the bits gain across the layers. Searching for the optimal interpretation, given Y, is equivalent to searching for the interpretation that yields the biggest bits gain. There are three components in $\log \frac{P(Y,I)}{P(Y,\phi)}$, the component $\log \frac{P(Y|X(I))}{P(Y|X(\phi))}$ is called data bits gain, the component $\log \frac{\prod_{\beta \in B}(\epsilon^{\beta}_{x^{\beta}}/\epsilon^{\beta}_{0})}{\prod_{\beta \in B(I)}(1-\epsilon^{\beta}_{0})}$ is called context-free (Markov) bits gain and the component $\log \prod_{\beta \in A(I)} \frac{p^{c}_{\beta}(a^{\beta}(I))}{p^{0}_{\beta}(a^{\beta}(I))}$ is called relational (non-Markovian) bits gain. The total bits gain serves as a fitness measure for each candidate object. Interpretation I could be a simple part, mutiple objects or a complex scene; the likelihood ratio treats them in the same manner. During implementation, candidate objects of different complexity are generated and stored with their bits gain.

4.1.4.4 Notes about Receptive Field

We borrow the word "receptive field" from neurophysiology, which says each neuron has a receptive field that represents the region of the visual field in which stimuli will affect its level of activity. In our setting, each brick has a RF just like a neuro and its RF is defined as the region which contains its all possible instantiations. In other words, each brick can be instantiated with its interest point being anywhere in its RF. That tells us each brick is sensitive to general activity (instantiation) in its RF, but insensitive (invariant) to the fine details of an activity (instantiation). As we build up the hierarchical structure (Markov backbone), higher layer bricks (representing more mature objects) naturally have bigger RFs, thus are more invariant than the lower ones. context-sensitive attribute is later imposed on each layer (in the form of non-Markovian ratio) to introduce selectivity. Then high selectivity and high invariance are both achieved for high layer bricks.

4.1.4.5 Notes about the Truncated Normal Distribution

Since each attribute in our case takes value in discret integer and its range is constrained by the "part" bricks' bounded receptive fields, we need to truncate (and renormalize) the normal distribution (to the same support as the null) to be able to conduct a fair comparison among different interpretations. Without this step, the likelihood ratio is unstandardized and can be misleading. We always consider distance in two directions (i.e., vertical and horizontal distance) separately, so we can keep our attributes as integers. Mean and variance of involved normal distributions depend on the label of the object.

4.1.4.6 Notes about Access Attributes

As described before, each object has a particular corner, which is used for computing attribute under composition. Also we have mentioned that the corner of each object is equivalent to (or defined as) the corner of one of its components at one layer below. So in a recursive manner, we can trace the corner of a high layer brick all the way down to the terminal layer, where we can read out its image location. That's the exact way we access attributes of high layer bricks. This implies that activation of a high layer brick needs supporting evidence from all layers below. Such a characteristic destroys the Markov property and complicates the dependency structure among bricks (for details, see §4.2).

4.2 Dependency Structure among Bricks (for license plate application)

Given that each brick is a random variable, taking values in its state space, we have a huge collection of random variables (abbreviated as RVs) from the constructed hierarchy. This section is devoted to exploring the dependency structure among all the RVs. The range of the connectivity between RVs determines the richness of the model, as well as the hardness of the problem.

Consider the prior model in Eqn 2.2. The dependency structure for the bricks can be formed in two steps, first through the Markov backbone, then further enriched by the non-Markov term. The Markov backbone is essentially a Bayesian net over the directed acyclic graph, whose dependency structure can be characterised as: each brick is linked to all its children bricks and bricks which share children bricks are linked. Non-Markovian ratio adds more connectivity into the dependency structure by linking relevant bricks at the same layer, or different layers if they would both occur in one of the possible instantiations of a license brick.

It is not feasible to draw the dependency structure described above among all the bricks; there is another intuitive way to determine whether a link between two RVs (bricks) exists. Given any two bricks, we first check their connectivity based on the Markov backbone; if they are not linked, we further check their connectivity based on the non-Markovain term.

Two bricks are linked due to the Markov backbone, when:

1. one brick is a child brick of the other brick.

2. they have common child brick(s).

Two bricks are linked due to the non-Markovian term, when:

1. they are in the same children set of a parent brick.

2. they both occur in one of the possible instantiations of a license brick. (This is the worst scenario, sometimes we may have conditional independence for certain pairs of bricks. Listing all the possibilities will be too cumbersome.)

In a word, this is a very rich model (equivalent to a few coupled Markov Random Fields). Many remote bricks are connected and the connectivity is quite dense. The only place that is not connected often is between string related bricks and boundary related bricks, because hieraries of string and boundary are constructed separately due to the fact that they do not share parts. However, string bricks and boundary bricks do connect to form license bricks.

4.3 Building a Compositional Machine

4.3.1 Data Model

The data model connects interpretations with the gray level image. We assume that given an interpretation, the data distribution only depends on the state configuration of the bricks on the terminal layer. That is,

$$P(\vec{y}|I) = P(\vec{y}|\{x^\beta : \beta \in \mathcal{T}\})$$

where $\mathcal{T} \subseteq \mathbf{B}$ is the set of terminal bricks. In order to be able to parse the scene robustly under different illuminations, some extent of photometric invariance is preferred. Rank sum statistic serves the purpose very well.

4.3.1.1 Rank Sum Statistics

For each terminal part, we have a template filter with entries "-1" (representing "black" pixels), "1" (representing "white" pixels), "0" (representing boundary pixels or pixels we don't care about). The black portion of each "part filter" represents n pixel locations that are expected to be dark, relative to m locations represented by the white portion of the filter. The appendix contains pictures of part filters. The rank sum R of the intensities of the corresponding "black" pixels, among the union of intensities of black and white pixels, is a convenient sufficient statistic that is demonstrably invariant to all monotone transformations of the image histogram. We model the intensities of the pixels (referenced by active terminal bricks) by assuming that their distribution depends only on R. That is,

$$P(\vec{y}(x^{\mathcal{T}})|\{x^\beta : \beta \in \mathcal{T}\}) = P(R(\vec{y}(x^{\mathcal{T}}))|\{x^\beta : \beta \in \mathcal{T}\})$$

where $x^{\mathcal{T}}$ refers to the state configuration of the terminal bricks, $\vec{y}(x^{\mathcal{T}})$ stands for the pixels referenced by $x^{\mathcal{T}}$ and $R(\vec{y}(x^{\mathcal{T}}))$ is the rank sum, computed out of those pixels.

We also assume that pixels that are not referenced by any active terminal brick are uniformly and independently distributed. For the detailed distribution of R, see next subsection. The rank sum statistic for each part filter is computed at each position of the image.

4.3.1.2 Three Components Model

Details of the Empirical Histogram

After we have collected these rank sum statistics, (i.e., for each template part, we have a matrix with similar size as the test image, whose entry stores a rank sum number), a histogram of the statistic is generated for each part filter.

The histogram consists of a central hump, two peaks at both ends and two shoulders in between. The shape of the histogram depends on the part filter and training images. Generally, the shoulder effect (i.e., heavy tail effect) comes from good-but-not-perfect match, the peak effect comes from perfect match, and the hump effect comes from constant patches. For filters with higher discriminancy level (i.e., bigger n and m, for details, see §4.4.2), the peak effect is less obvious and the shoulder effect lasts longer. Histograms of the bar filters with different discriminant levels are listed in Figures 4.7-4.14 (for pictures of the bar filters, see §4.4.2). Note: the horizontal scales (which represent the range of rank sum statistics for filters with different discriminancy levels) of the histograms are quite different.

Three-component Model and Justification

We model the above histogram with a mixture of three component distributions (one normal and two exponentials). The first exponential distribution (more precise, geometric distribution over the finite range of R) models the pattern of rank sum statistic R given that the target is present, and smaller R implies better fit. The second exponential distribution is introduced to better fit the roughly symmetric histogram by capturing the pattern of the opposite filter in the training set. The normal distri-

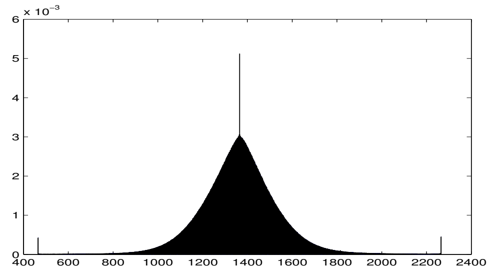


Figure 4.7: Histogram of level-1 bar filter

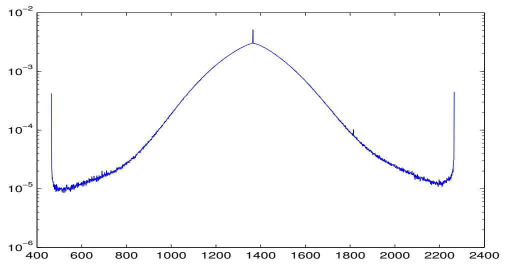


Figure 4.8: Its log plot

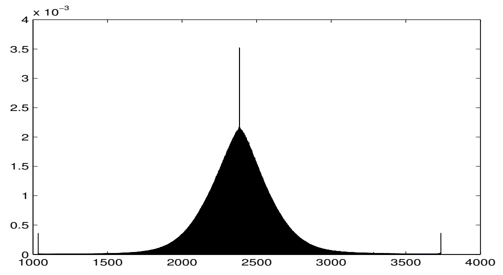


Figure 4.9: Histogram of level-2 bar filter

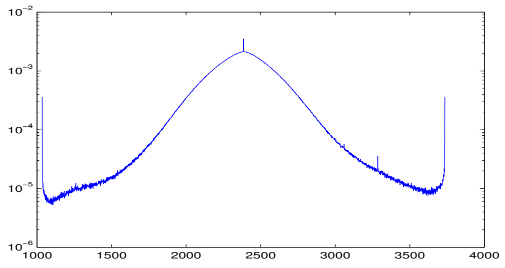


Figure 4.10: Its log plot

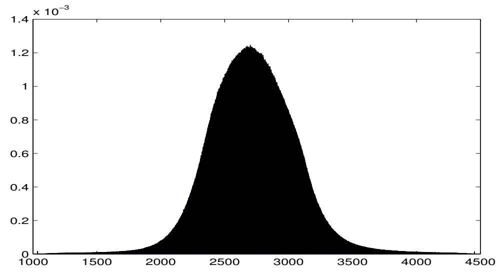


Figure 4.11: Histogram of level-3 bar filter

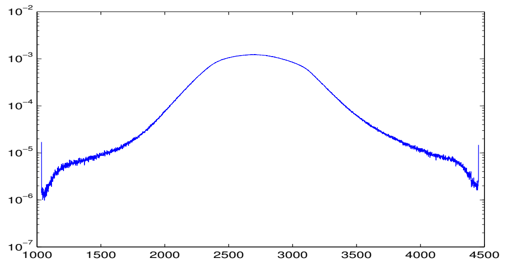


Figure 4.12: Its log plot

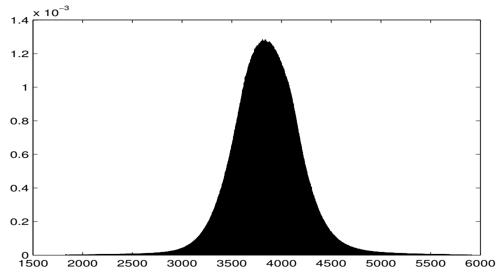


Figure 4.13: Histogram of level-4 bar filter

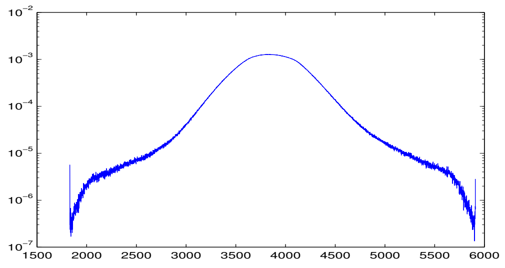


Figure 4.14: Its log plot

bution (more precise, truncated normal distribution over the range of R) models the pattern of R when nothing is present, which is justified by the central limit theorem from the assumption that non-referenced pixels are uniformly and independently distributed. The mean and variance of the normal distribution are functions of (m, n). (for more details, refer to [18])

It would be nice if we could fit the histogram perfectly, but given the free parameters we have, it is hopeless. Then why not try more components to add more parameters? The next reasonable model could have five components, with two exponential distributions explaining the peak ends (perfect match) and two other components explaining the shoulders (good-but-not-perfect match). But when we compute the ratio of probabilities of the part being there versus null given image Y, the numerator should include both the peak-end component (coming from perfect matches of the true part) and the shoulder component (coming from not-bad matches of the noisy true part or any similar part). Then the resultant five-component model produces a similar ratio to our three-component model, with little improvement in the fitting. Another main reason not to try a five-component model or even more complicated models is due to a property called "universality" (see below). This key property makes our computational scheme feasible. More free parameters will have less of a chance to achieve universality. It turns out that the simple three-component model can achieve this property while producing a good-enough fitting.

Universality

Let's step back a little bit. What do we really want out of this data model? The big picture is: we first conduct parameter learning through training for each part filter (building block); then we compute the likelihood ratio of its presence versus null for a fixed test image using learned parameters; finally we will compose suitable parts into a bigger object if the object gives us a larger likelihood ratio of its presence versus null than the joint likelihood ratio of its otherwise independent parts. (Note: probability of null is necessarily used in the ratio for a fair comparison.) This process is iterated up to the top layer. The key question during the process is how to compute the data likelihood ratio (or data bits gain) for a composed object. We cannot simply multiply the data likelihood ratios of its composing parts, due to the issue of overlapping. And we cannot compute it directly as we did for a terminal part, because it is infeasible to estimate the relevant parameter for a mature object (i.e., λ in the exponential distribution) through training, since there are too many ways to instantiate such an object. We need a property that says parameter λ of an object is independent of its shape. In other words, we need λ to be a function of (n, m) (i.e., total number of "-1" and "+1" positions) only. This is what we called "universality". Our goal is to find a data model which fits training data reasonably well and gives us universality.

Model Fitting

Given that we settle with the three-component model, there are four free parameters, ϵ_0 (weight of normal), ϵ_1 (weight of first exponential), λ_1 (in first exponential) and λ_2 (in second exponential) to estimate. The probability of R can be written as

$$\begin{aligned} P(R) &= P(X=1)P(R|X=1) + P(X=0)P(R|X=0) + P(X=-1)P(R|X=-1) \\ &= \epsilon_1 \cdot \lambda_1 e^{-\lambda_1 R} + \epsilon_0 \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(R-\mu)^2}{2\sigma^2}} + (1-\epsilon_0-\epsilon_1) \cdot \lambda_2 e^{-\lambda_2 R} \end{aligned}$$

where $X = 1$ refers to the presence of the part, $X = -1$ refers to the presence of the opposite part and $X = 0$ refers to the absence of any part. To be more precise, the above normal and exponential distributions are normalized over finite intervals. A few notes about how to fit are listed below.

1. The normal component is fixed with $\mu = \frac{n(n+m+1)}{2}$ and $\sigma^2 = \frac{n((n+m)^2-1)}{12}$, that is, the mean and variance for R given that nothing is present. We never fit the data with variable normal parameters, otherwise the fitted normal distribution will take over some of the exponential part. The finite interval for R is from $\frac{n(n+1)}{2}$ to $\frac{n(n+1)}{2} + nm$.

2. Ideally, we should apply EM to estimate all these parameters. But since the exponential (geometric) distribution is defined over some finite interval, there is no explicit formula for λ in terms of $E(R)$. The numerically estimated exponential component by EM (from the messy formula over the finite interval) is almost uniform (i.e., $\lambda = 0$). Since the three-component model is doomed not to generate perfect fitting and EM just finds a local optimal, we have already sacrificed accuracy. Why don't we just assign a simple form to λ, given all its justifiable properties? That is, λ has to be a positive, decreasing and symmetric function of n and m (the decreasing property comes from the long, flat shoulder effect as a filter becomes discriminant and the symmetrical property comes from the symmetry of the histogram). The simplest form with these properties is $\frac{C}{nm}$. (Note: $\frac{C}{n+m}$ also satisfies the above properties, but it turns out to be improper. For more evidence, see below.) There is another way to show the properness of such form: assuming $R \in [\frac{n(n+1)}{2}, \infty]$ (approximately true for big n and m), we have $\frac{1}{\lambda_1} = E(R) - \frac{n(n+1)}{2}$ under the first exponential component; if we further assume the first exponential component is quite flat over the long interval, then we will have $E(R) \approx \frac{n(n+1)}{2} + \frac{nm}{2}$; therefore, λ_1 has the same form. Most importantly, given that we don't assume any form, the estimated λ for each part filter by EM is very close to such form with a fixed C (between 1 and 2). This further validates the form for λ.

For general detection we use the most discriminant filter. A fitting of the histogram in Figure 4.13 is shown in Figure 4.15, which is generated by setting $\lambda_1 = \frac{1}{nm}$ and estimating ϵ_0, ϵ_1 and λ_2 by EM. As you can see, the fitting is not perfect, but the estimated ϵ_0 could give us some hint about the initial setting of the state probability of the bar brick (Recall in §4.1.1.1, $\epsilon_0^\beta = (\epsilon_0)^{25}$ and $\epsilon_i^\beta = \frac{1-(\epsilon_0)^{25}}{25}$ for i=1..25, where ϵ_i^β is the probability of terminal brick β taking value i).

Advantages and Disadvantages

The three-component model is one of the many possible candidate data models which achieve our goal.

Its advantages are:

1. It is the simplest data model which achieves good fitting and universality.

2. The data bits gain (i.e., $\log \frac{P(R|X^\tau(I))}{P(R|X(\phi))}$) is well approximated by a function of n, m and R. Under perfect match (i.e., $R = \frac{n(n+1)}{2}$), such function has some nice properties. They are continuous and roughly additive when n and m are close.

Its disadvantages are:

1. Not a perfect data model in terms of fitting.

2. Doesn't differentiate between perfect match and good-but-not-perfect match. It contains potential risk of false positives.

4.3.1.3 Implementation Details

Part Filters and Their Interest Points

Bearing reusability in mind, we manually designed 47 part filters to cover all the characters and the license boundary. We tried four sets of character parts with different discriminancy levels (details in §4.4.2). We generally use the most discriminant

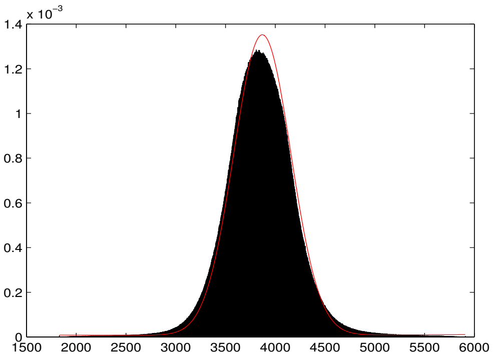


Figure 4.15: histogram fitting

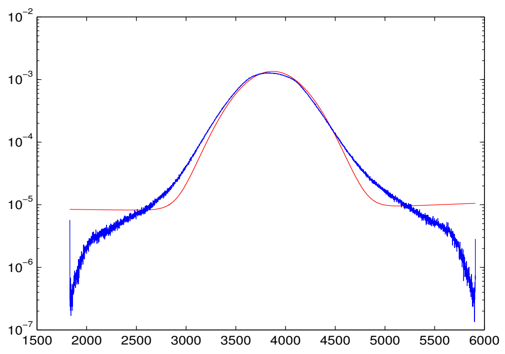


Figure 4.16: Its log plot

set unless otherwise specified (for pictures of them, see Appendix). Each part filter has one interest point, which is the upper-left corner for a character part and either upper-left or lower-right corner for a boundary part. Each higher layer object has one interest point only, the object's upper-left (or lower-right) corner, which is defined as the interest point of the object's upper-left (or lower-right) part.

Principles for Designing Part Filters

A few design principles are listed below. They are independent of any training/test image.

1. Make the part filter(s) dense at those positions where fine detail matters. Say, the upper part of "0" and "D." (The lower part is just a flip of the upper part in both cases.) The detailed difference between the curvy corner of "0" and the perpendicular corner of "D" is captured by dense filters.

2. Put contrasting pixels at the proper places for similar parts. Say, the upper part of "3" and "5." Besides the "-1" positions (on the stroke) and "+1" positions (neighboring the stroke), we fill in "+1" in the upper part filter of "3" at places where the vertical stroke of the upper part of "5" stands and fill in "+1" in the upper part filter of "5" at places where the 45 degree stroke of the upper part of "3" stands, instead of leaving them at 0. (See pictures of part 8 and part 12 in Appendix.)

3. The numbers of "-1" and "+1" positions devoted to one unit length of similar strokes should be similar to each other. Thus, under perfect matching, similar parts have similar bits gains and parts with a lengthy stroke (i.e., a larger dark region accompanied by a larger bright region) have bigger bits gains.

Assumption to Simplify the Data Model

The original data model should be pixel-based, connecting each gray-level intensity to a hidden binary variable for each terminal part representing its absence/presence. This is compatible with three-component EM fitting in §4.3.1.2. As noted in §4.1.1.1, instead of a pixel-by-pixel mapping between image space and terminal layer, each terminal brick covers a 5×5 non-overlapping image patch. The simplifying assumption here is: there are no multiple parts of the same type in each patch. In other words, the best fit among the patch (i.e., the smallest ranksum out of 25 numbers) decides absence or presence (and the location) of each part. This is the major approximation introduced in our model. The patch size (5×5) is proper for this application and subject to change for others. Note: multiple parts of different types in each patch are allowable.

Rule for Forming Object/Scene by Combining Terminal Parts

The data term of an object/scene (in the rank sense) is represented by a set of "-1" positions and a set of "+1" positions, which is formed by a majority vote of its relevant terminal parts. In other words, we cast the terminal part filters into the image space, according to the state configuration of the terminal bricks in a particular I; then for each pixel location, we sum all the votes ("-1" and "+1") from possible overlapping part filters. If the sum is negative, that position is added into the "-1" set of this object/scene; if the sum is positive, it is added into the "+1" set. Locations with sum zero are ignored. Rank sum R computed based on the "-1" and "+1" sets is the sufficient statistic in the data model for the object/scene. With the help of universal λ, we can compute the data likelihood ratio of an object/scene without explicitly fitting its rank sum, unlike what we did for the terminal parts.

Terminal parts in an I can be disjoint or overlapping, but this rule treats them the same. Also, as we can see, by storing the votes at each position during the parse, this rule guarantees that the probability of a parsed scene is independent of its parsing order such that the data model is well defined.

Implementation of the Data Likelihood Ratio of an Object/Scene versus Null

Given the independence assumption of Eqn 2.3 in §2.2, the data likelihood ratio (of an object/scene versus null) only depends on the state configuration of bricks at the terminal layer, that is, $\frac{P(R|X^{\tau}(I))}{P(R|X(\phi))}$. Based on the three-component model for R (in §4.3.1.2), we know the numerator of the ratio refers to the first exponential component and the denominator refers to a mixture of a normal component and the second exponential component. Since the second exponential component is negligible compared to the normal component (when R is less than the mean under the null), but imposes a heavy computational burden (further worsened by frequent calculations of the ratio), we drop this component in the implementation. So from now on, the data likelihood ratio is approximated by the ratio of the first exponential component versus the normal component.

Current Parameters in the Implementation

During implementation, instead of applying EM or empirical counting to estimate state probabilities for bricks (as suggested in §4.1), we simply set the null probability ϵ_0^{β} for a character part brick β to be 0.99 or 0.9999, depending on its size (there are only two sizes, the smaller one has $\epsilon_0^{\beta} = 0.99$), and for any boundary part brick to be 0.999; for any character brick to be 0.99; for any partial string brick to be 0.999; for any string brick to be 0.99, etc. Each active state probability ϵ_i^{β} of brick β is equal to $(1 - \epsilon_0^{\beta})$ divided by the number of brick β's nonzero states. λ takes the form of $\frac{1}{nm}$. Under such a setting, the perfect-match data bits gain for individual characters under the most discriminant filters ranges from 100 bits to 150 bits with less than 1 bit per pixel. After the initial parameter setting, we never change them.

As we will see later, even with such carelessly specified parameters, we still achieve great performance. This is because the hardwired hierarchical structure (in §4.1) and the introduced relational binding (in §4.3.2) indeed play a key role. It is always possible to further improve our results by careful parameter learning (from EM and empirical counting).

4.3.2 Relationship

As mentioned earlier, a composition system is generated through a sequence of perturbations on its Markov backbone. Each perturbation introduces a relational binding of a certain attribute into the structure. In other words, it is relationship (in the form of non-Markovian ratio) that differentiates composition system from its Markov backbone. Implementation of this part usually involves a fair amount of modeling, such as designing compositional rules and specifying rule-dependent probability distribution on relationship. Before we get to the modeling part, let us first justify its role and see a simple example.

4.3.2.1 Motivation and a Simple Example

A legitimate object is defined as a composition of its constituent parts. Evidence for the composition depends on the appropriate details of the instantiations of its parts. Under the Markov backbone structure, there is no restriction on the details of the relative arrangement of an object's composing parts. The samples from such a distribution produce interpretations that have, at each layer, the right pieces but in the wrong arrangements. (See demonstration in §4.4.1)

Under the perturbed distribution (i.e., product of Markov backbone and non-Markovian ratio), we overcome the shortcoming of broad coverage by introducing a likelihood ratio that explicitly measures the suitability among the composing parts that instantiate an object in a particular interpretation. Such a ratio is a function defined over a collection of attribute functions involved in an applicaton. In our license demostration system, typical attribute function measures the relative distance of interest points of relevant composing parts. For example, the attribute function for a 2-letter brick measures the horizontal and vertical distance between the upper-left corners of the two letters.

4.3.2.2 Design of Compositional Rules

Define the Support

Equipped with the context-free rules (in Appendix), to form a compositional rule, we first need to specify the support of each attribute. Recall the way we form children sets for a non-terminal brick: we fix its first part, allow its other parts to float around a neighborhood to achieve robustness, and then we move the first part around a neighborhood (along with other parts) to achieve translation invariance. All the covered combinations constitute a collection of children sets. In other words, both robustness and translation invariance of an object are taken into account when hardwiring its children sets. However, for the support of a compositional rule, only the robustness matters. For example, for a two-part object, the support of relative x-distance among the object's constituent parts is an interval with its boundaries computed by taking difference between two closest end points and two farthest end points along x-direction from the RF (RF as defined in §4.1.4.3) of its fixed first part and union of RFs of its floating second part; for a three-part object, apply the same procedure twice for each of its neighboring two parts; etc. Given the support, we can start to specify distributions on them.

Compute the Relational Distributions under Composition and under Null

For every attribute a associated with brick β, we craft a compositional distribution p^c_β and estimate a null distribution p^0_β. p^c_β, describing the 'true' relationship among constituents of β under composition, can be set as we want; but p^0_β is usually not available, either in closed form or through Monte Carlo estimation. When there is no confusion, we will drop the subscript β in the discussion.

Fortunately, in our setting p^0 can be approximated under reasonable assumption. Given the hardwired hierarchical structure, it's not hard to compute the RF (of the interest point) of each brick at a different layer. Assume the interest points of two composing parts take position along the x (or y) direction uniformly and independently over their allowable regions (from which the support is computed), the distribution p^0 of the relative distance along the x (or y) direction is triangularly distributed, which can be derived by exhaustive enumeration all the possibilities. From the way p^0 is generated, for any context-free sample under our Markov backbone, no matter how far its parts are separated, p^0 over any of its attributes is always positive.

For simplicity, p^c is set to be a truncated normal distribution (with the same support as p^0) for each attribute, whose mean and standard deviation can be estimated (or adjusted) conveniently. For a fixed object, the mean of its attribute can be measured with high accuracy and the standard deviation of its attribute is guessed initially depending on the complexity of the object (a greater standard deviation is assigned to an attribute of a more mature object to enhance the robustness). After training on a few images, we make some adjustments to the standard deviations and never change them afterwards. (Note: there is no potential overfitting here. Altogether less than 10 different standard deviations are involved, since attributes are widely shared among different objects.) The ratio of p^c and p^0 gives us a signal for composition against chance.

p^c and p^0 for each attribute are computed once and stored. In the implementation, a candidate object (brick) β is formed when the likelihood ratio of its attribute (only involved for that layer) $\frac{p^c_\beta(a)}{p^0_\beta(a)}$ passes a fixed threshold. The reason we don't threshold the whole likelihood ratio of composition versus non-composition, like $\frac{\epsilon^\beta_i/\epsilon^\beta_0}{\prod_{\alpha \in C^\beta_i - B(I)}(1-\epsilon^\alpha_0)} \times \frac{p^c_\beta(a)}{p^0_\beta(a)}$ as suggested in §4.1, is due to the sufficiency of the relational constraint and lack of confidence in state probability estimation.

4.3.2.3 Potential Generalization

Current string-related attributes we are applying include the relative distances between two parts (for a 2-part character), between two partial overlap neighboring parts (for a 3-part character), between two neighboring characters (for a partial string) and between two partial strings (for a string). Current boundary-related attributes that we are applying are the relative distances between two parts (for a long line), between two perpendicular lines (for a L junction) and between two L junctions (for a boundary). Current attribute for the license is the relative distance between boundary and string.

For our fixed font license plate, the above attributes provide an excellent binding. To deal with test images with more ambiguity, the attributes can be generalized (improved) in many ways. A few of them are listed below:

1. Instead of considering relative distance attributes only between neighboring parts and independently, consider a joint normal distribution over all relevant relative distances.

2. Introduce more interest points for each object and more bindings among them. Currently we only have one interest point for each object (i.e., one of its corners). We could include its other corners or centroid as well.

3. Extend the support of the attribute function for a viable object (say high-layer objects) to strengthen the robustness.

4. Craft p^c for each attribute with its own distribution (this may not be normal-like, as any distribution with a thinner tail than the triangular distribution is feasible) and set its own threshold. Currently, p^c is always a truncated normal distribution and we only use one threshold for combined attributes (say, one threshold for product of likelihood ratios of attributes d_x and d_y). In a real situation, we could consider assigning a non-normal distribution to p^c for attribute in some too-often-occuring objects (like "L") and we could use two thresholds for likelihood ratios over d_x and d_y separately, since some object's x-direction attribute is more flexible under different illuminations (say, partial string).

In short, dense and redundant binding plus flexible modeling and thresholding can further improve our performance. Some evidence learned from examples can be found in §4.5.

In a more general setting, besides relative distance, attributes like relative scalar, rotation, colorness, etc., can be introduced accordingly.

4.3.3 Scene Parsing

4.3.3.1 Ambiguity Propagation and Pruning

In the bottom-up pass, conservative thresholds are applied for lower-layer bricks to elicit possible activities among high-layer bricks. In the meantime, contextual information (at a higher layer) is used to get rid of the poorly composed candidates (propagated from a lower layer). There is information flowing between lower level and higher level.

Implementation Details

For each brick in the above set (i.e., non-terminal brick), we need to check the collection of all its children sets. Note: an active brick in the above set can have multiple candidates. After we generate the candidates for each brick at a lower layer, clustering and pruning are applied on each of them. We only keep the top N candidates of each active brick for further (similar) process at the next layer. String-related objects and frame-related objects are generated independently through their own hierarchies. License object is formed by combining string object and frame object. Two major functionalities are involved here: one is candidate generation (also called list generation) and the other is clustering and pruning.

Detailed Description for List Generation

Given a fixed test image, after computing the rank sum matrix for each terminal part, we compare the best candidate (smallest rank sum) out of each non-overlapping 5×5 patch in the matrix of that type with a threshold (depending on the type). All the parts that pass the simple test (i.e having smaller rank sum than the threshold) enter into the list, and their corresponding bricks are set to be on. Usually we set the threshold to be quite large to allow more poor parts to survive the test. Note: this test is equivalent to testing the likelihood ratio mentioned in §4.1.1.1, because $\frac{P(R|X^{\tau}(I))}{P(R|X(\phi))}$ is a monotone decreasing function of R, for R less than the mean under the null.

For each brick on the second layer and above, we check the collection of all its children sets. Once a set of its children bricks is active, we compute their non-Markovian ratio and compare it with a threshold (depending on layer and type). If the ratio passes the threshold, we activate such a brick and store its index (including layer, type, location, etc.) and the pointers to its children bricks into the list. Through the pointers, the subgraph of the candidate object can be traced (Note: every object is a subgraph in our representation). One active brick can have multiple candidate objects, due to its wide receptive field.

String-related (character, generic character, partial string, string) and frame-related (long lines, L junctions, frame) candidate objects are generated separately into the same list by the above process, because their parts don't share subparts. The license plate brick on the top layer has string brick and frame brick as its children set.

The list generation process can be quite time-consuming because of the conservative thresholds for the lower layers. This is the reason why we introduce a clustering and pruning scheme for the bricks of higher layers.

Detailed Description for Clustering and Pruning

Clustering scheme is applied during the intermediate layers of the hierarchical structure to compress the potential candidates in order to propagate the ambiguity efficiently. The scheme we implemented is layer-dependent.

For each brick on the generic character layer, we cluster its candidates by their states. For example, a generic digit brick can have candidates with states ranging from 1 to 10. Due to a relatively small receptive field, clustering only by states at that layer serves the purpose well. The number of clusters and size of each cluster are adjustable parameters. Candidates not assigned to any cluster are thrown away from the list.

For each brick on the other layers (partial string, string, long line, L-junction, frame), we cluster its candidate by the positions of their interest points. For example, a 2-letter brick can have multiple candidates with their interest points (i.e., upper-left corners) ranging all over its receptive field. The center of the first cluster is where the best candidate in that brick is located; the center of the kth cluster is where the best candidate in that brick outside the first k clusters is located. Cluster radius for each layer is predefined and varies over different layers. Each candidate can be assigned to a cluster at most once. Again, the number of clusters and size of each cluster are parameters to control the extent of clustering and pruning.

For license detection purpose, we do not further cluster candidates at the license layer. The top N candidates for each brick of this layer are chosen into the list. There is no explicit clustering for terminal layer either, but picking the best candidate out of a 5×5 patch is like an implicit clustering.

4.3.3.2 List Optimization

Ideal List Optimization

The generated list includes local interpretations that are largely redundant, differing only in the detail of positioning, as well as others that are mutually inconsistent. This is the index set, a set of objects with different complexity that we next employ in a greedy algorithm to compute a full-blown parse. The best candidate (measured in bits gain) is chosen from the list to start the parse. Then, conditional on the already chosen object (subgraph in compositional architecture), we search for the candidate that most increases the joint bits gain of the parse when combined with the already-selected one, from all consistent candidates in the remaining index set. The pair of consistent subgraphs defines a new parse with bigger bits gain. We continue this procedure until no more candidates from the index set can increase the total bits gain. The output configuration is the first full-blown interpretation of the test image. We can get more interpretations by starting over with different objects, say the next-to-best object. Everything else follows exactly the same idea. After we collect quite a few interpretations (parses), the best one (with the biggest bits gain) is chosen as the final interpretation.

Since this is a greedy algorithm, we may not necessarily achieve the optimal interpretation of the test image. But with coherent implementation of the compositional rules and non-Markovian ratios, this algorithm does provide us with a platform for a fair comparison between candidate objects. Also, the bottom-up search strategy generates a good index set, which narrows the search space enormously, and scene covering further employs top-down information, so we can find a good approximation for the covering problem.

Pseudo List Optimization

In practice, we use pseudo list optimization for fast processing. Note that through the way we generate the list (see list generation above), it could possibly contain inconsistent objects (subgraphs). For details of inconsistency, see below. Also, the bits gain for each candidate object is calculated as the sum of the bits gain of its subparts and the bits gain from composition. We call this "pseudo" bits gain. This is not very precise because we didn't deal with the overlap region properly (precisely, the data term in the overlap region covered by a candidate object's parts is double counted). We could either correct them (i.e., inconsistency and pseudo bits gain) when we are generating the list, or we can postpone them to the list optimization stage. It turns out that the latter scheme has great computational advantage. We adopt the latter scheme, and the final stage is called pseudo list optimization. That is, given a list of generated candidates that may or may not be consistent, we pick the best consistent object in terms of "pseudo" bits gain and compute its true bits gain (see "true bits gain computation" below), then conditioned on the already chosen object, we do some pruning over the list (see "list pruning" below), then find the best consistent candidate in the remaining list (see "consistency checking" below) and finally, compute the joint true bits gain (i.e., overlap region is counted only once) of the already chosen object and the current best one (see "true bits gain computation" below). If the overall bits gain increases, the current best one is selected for inclusion into the interpretation; otherwise, it is deleted from the list. The process iterates over the sorted list once and stops. The output gives you a full-blown interpretation. Similarly, we can have multiple interpretations with different starting objects.

This optimization scheme is always inferior to the ideal one. However, it runs much faster and yields competitive results. The tradeoff between performance and computational resource is worthwhile.

Implementation Details

Three major functionalities (consistency checking, true bits gain computing, and candidate pruning) are discussed in this part of implementation.

Consistency Checking

There are two kinds of consistency checking. One is within an object and the other is between the chosen object and the current configuration. By the philosophy that an object's subparts are also objects, we can deal with the two kinds of checking together.

Consider two active bricks that share a common child brick which has many possible instantiations; each of the two active bricks points to one instantiation of the common brick.

Case 1: The instantiations of the common brick pointed by the two active bricks are different (i.e., under the two different instantiations, the common brick has different states, or different attributes, or different instantiations of its parts, or equivalently, different subtrees). We term such a configuration inconsistency.

Case 2: The two instantiations of the common brick are the same (i.e., they share the same subtree). We term such a configuration consistency.

Once we achieve consistency, the context-free ratio for that common child brick is only counted once, as in the Markov formular (see "Below set" in Eqn 2.1). The Non-Markovian ratios for the two parent bricks are counted separately.

True Bits Gain Computation

Given the current best consistent object, which is also consistent with current configuration (i.e., already chosen objects), we apply the majority voting rule to get updated sets of "-1" positions and "+1" positions for the new configuration (like the rule for forming object from combining terminal parts in §4.3.1.3). Then, universal λ is computed by using the number of updated "-1" and "+1" positions. Finally, the true bits gain (with respect to the null interpretation) is computed by summing the data bits gain (from the data likelihood ratio), context-free bits gain (from the likelihood ratio of Markov backbone) and relational bits gain (from the Non-Markovian ratio).

List Pruning

After we add one more object into the current configuration, we prune the list to narrow the search space. The general pruning principle we apply here is that once a brick is occupied, any candidate brick at the same grid of other sublattices on the same layer is not allowed, except that the occupied brick is string-related and candidate bricks are boundary-related, or vice visa. Since a complicated object, whose subtree covers a wide range of bricks at different layers, is often chosen at an early stage, lots of candidates at different layers are pruned very quickly.

4.4 Demonstration of the Themes

4.4.1 Markovian versus Non-Markovian Simulation

Message: Under the Markov backbone, a typical sample (also called context-free sample) is poorly composed of its parts, due to its broad coverage, while under the compositional distribution including the non-Markovian term, a typical sample (also called context-sensitive sample) is more properly composed of its parts.

Brief description of the demonstration: We demonstrate this message through generating a few 4-digit partial string samples under the two distributions (with and without the non-Markovian term). With the framework we have set up, a context-free 4-digit sample can be generated very quickly. We apply the idea of sequential importance sampling (instead of one-step importance sampling) to generate a typical context-sensitive 4-digit sample in a reasonable time.

Detailed description of the demonstration: To generate a context-free 4-digit sample, we start with activating a fixed 4-digit brick on the partial string layer by randomly picking a state according to its state probability vector, conditioned to itself be on. This leads to its selected four children bricks being on. Next, we independently choose the states of the selected children bricks in the general digit layer, conditioned to themselves be on. Since we are only interested in the instantiations of a 4-digit brick, we ignore bricks that are not relevant. Continuing downward, we finally choose the states of the selected terminal bricks. This procedure generates a complete subgraph in a Markov manner, given that the scene consists of one 4-digit object. A few context-free samples are listed in Figure 4.17.

It is straightforward to generate a context-sensitive 4-digit sample by one-step importance sampling. That is, first generate N context-free 4-digit samples by the same scheme described above and store them; next reweigh each sample by its non-Markovian term; finally generate a context-sensitive 4-digit sample by their weights.

Under such a scheme, N has to be very big to generate a well-posed 4-digit sample.

To generate a context-sensitive 4-digit sample more efficiently, sequential importance is applied. A few steps are involved:

1. Generate N general digit context-free samples by a similar scheme as above and store them.

2. Reweigh each digit sample according to the quality of relationship between its composing parts by the non-Markovian term.

3. Randomly choose a state for a fixed 4-digit brick, given it is on. At the positions of its selected children bricks, pick 4 previously stored general-digit samples according to the new assigned weight. Reweigh such a 4-digit subgraph using the joint non-Markovian terms, which involve attributes within each digit and between the four digits.

4. Repeat step (3) for M times and store all the subgraphs and their weights.

5. Generate a context-sensitive 4-digit sample from the stored M subgraphs, according to their weights.

We run the above process with $N = 10000$ and $M = 10000$ twice. We list the top three context-sensitive samples for both runs in Figure 4.18.

Even though the context-free samples are quite "disfigured," we can still read them from Figure 4.17 ("4850," "3034," "8437" for the top row and "2648," "6240," "6809" for the second row), due to the limited binding from the Markov backbone. The top three context-sensitive samples in Figure 4.18 have relative probabilities (with respect to the top one, i.e., the leftmost one) 1, 0.1178 and 0.0099 for the first run and 1, 0.4984 and 0.3964 for the second run respectively, which look more properly composed. We could have even "nicer" samples if even stricter relational binding were applied (by replacing current σ in p^c with a smaller one to penalize poor composition).

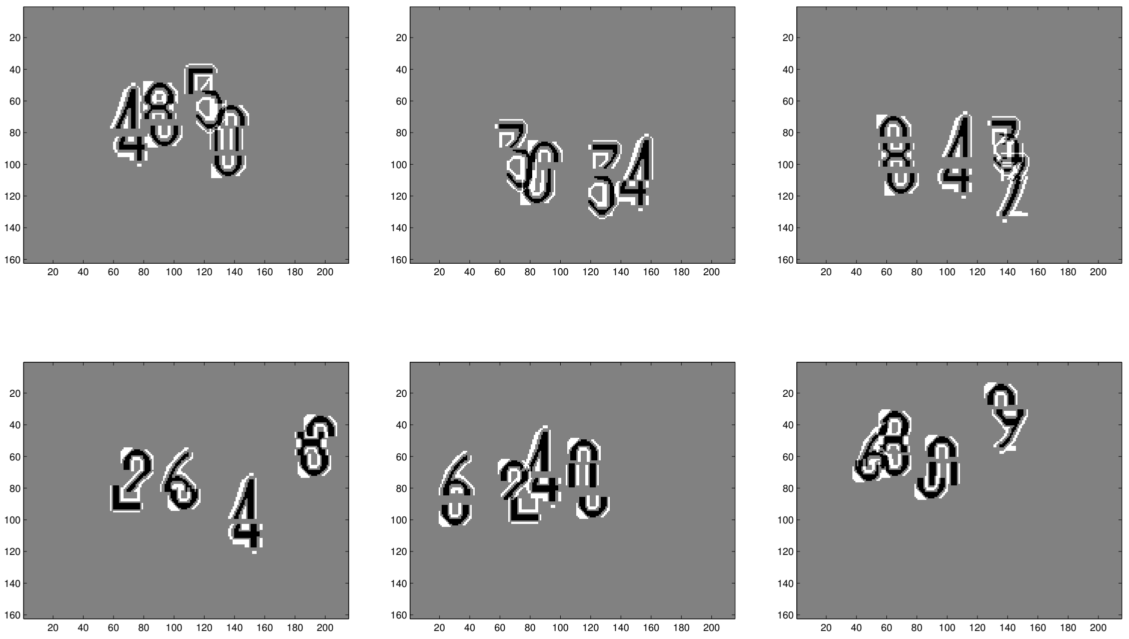


Figure 4.17: context-free samples

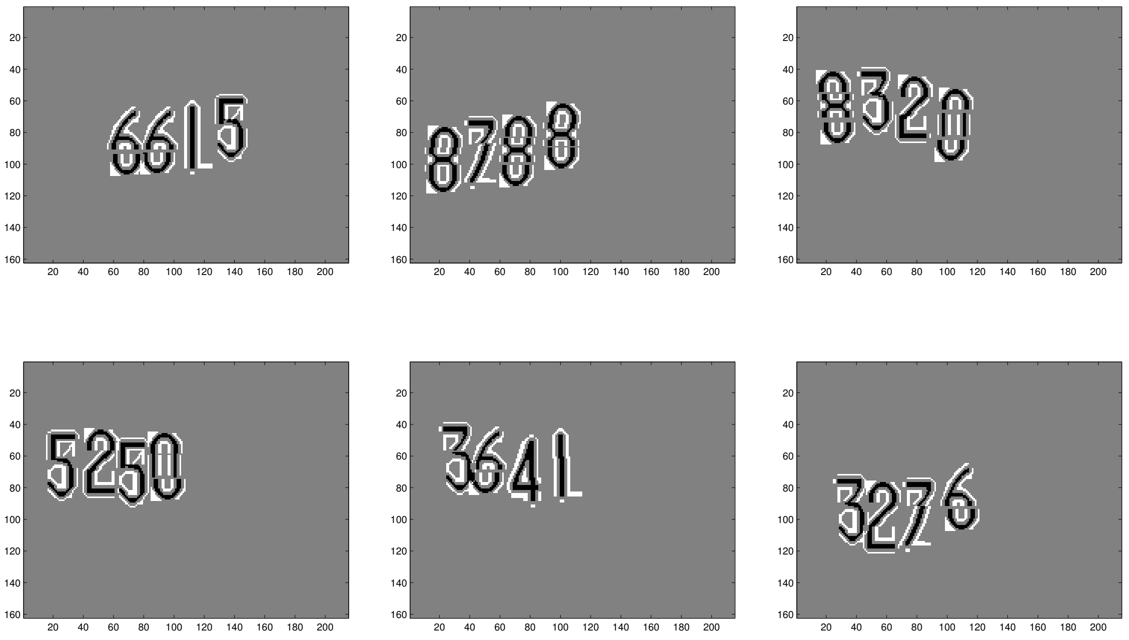


Figure 4.18: top three context-sensitive samples (in upper row and lower row) for the two runs

4.4.2 Effects of Data Term with Different Discriminant Levels

Message: Data term matters in the parsing setting. It can change the interpretation dramatically even if there is already a well-defined relational binding.

Brief description of the demonstration: We demonstrate this message through parsing a few test images using part filters of different discriminancy while keeping everything else unchanged. The more discriminant filters we apply, the finer details of the characters we can capture. Similar objects' subparts often share a common relationship; it is thus necessary to apply discriminant filters to differentiate them.

Detailed description of the demonstration: Four sets of part filters (for the characters) with different discriminant levels are applied. The set of part filters with the least discriminancy (most robustness) describes each straight stroke as a two-pixel-thin black portion in contrast to its neighboring two-pixel-thin white portion within a small box (15 by 20 for a smaller part or 25 by 20 for a bigger part), with a two-pixel gap in between. The set of part filters with next-to-least discriminancy describes each straight stroke as a three-pixel-thin black portion in contrast to its neighboring two-pixel-thin white portion within the same small box, with the same gap. The set of part filters with next-to-most discriminancy describes each straight stroke as a three-pixel-thin black portion in contrast to its neighboring two-pixel-thin white portion within a big box (23 by 26 for a smaller part, or 33 by 26 for a bigger part), with the same gap. The set of part filters with the most discriminancy (least robustness) describes each straight stroke as a four-pixel-thin black portion in contrast to its neighboring two-pixel-thin white portion within the same big box, with a one-pixel gap. This is the general rule when we design different sets of filters. After that, extra care has to be paid to follow some design principles in §4.3.1.3. For simplicity, we call them level-1 (least discriminancy), level-2, level-3 and level-4 (most discriminancy) filters. An example of a part filter with different discriminant levels is shown in Figures 4.19-

4.22. For a fixed test image, we parse it using four sets of filters separately while keeping everything else unchanged. We only output the top object for comparison purpose. Note: parts of the license boundary have fixed discriminant level.

For our fixed-font license plate detection application, it is easy to see that the more dense part filters we apply, the fewer false positive parts and objects we get, and the more power we have to differentiate similar characters. This does require more time to be spent on the computation of rank sum statistics of the dense filters. Some parsing results under different discriminant filters are shown in Figures 4.23-4.40.

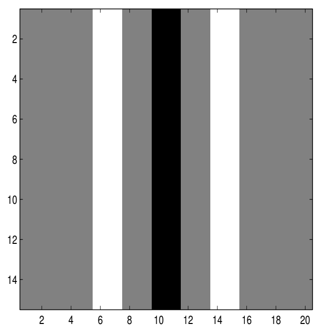


Figure 4.19: Part 4 of level-1

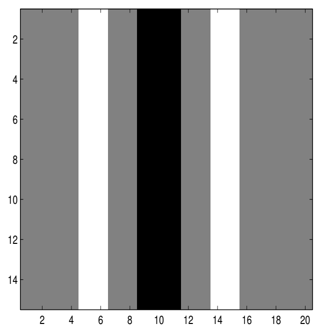


Figure 4.20: Part 4 of level-2

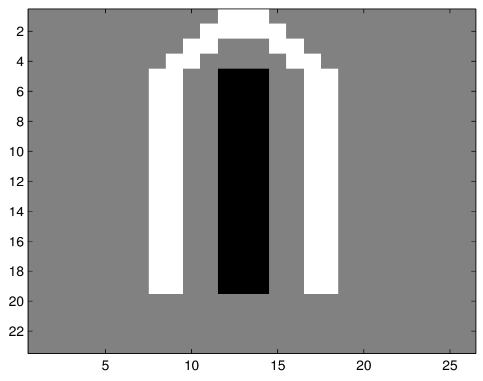


Figure 4.21: Part 4 of level-3

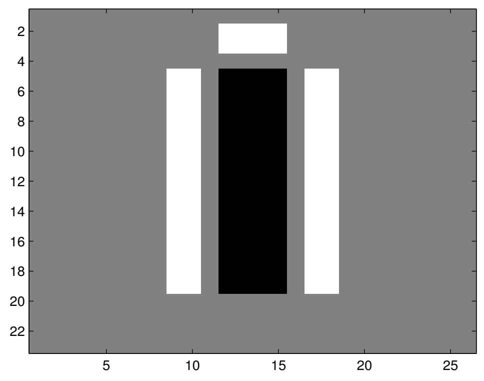


Figure 4.22: Part 4 of level-4

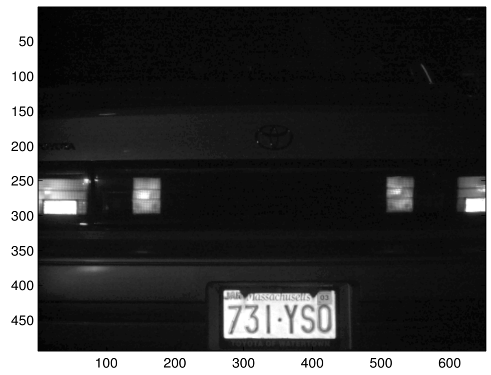


Figure 4.23: original image

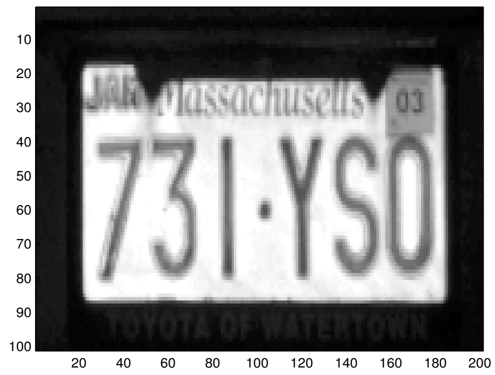


Figure 4.24: zoomed license region

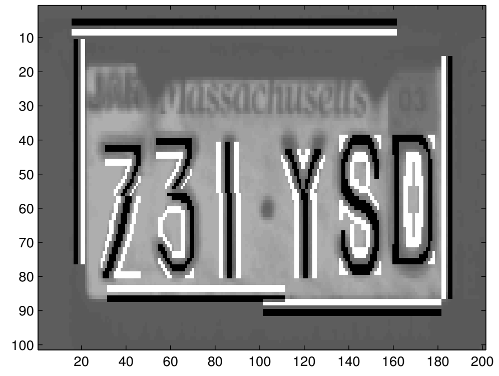


Figure 4.25: top string by level-1 filters

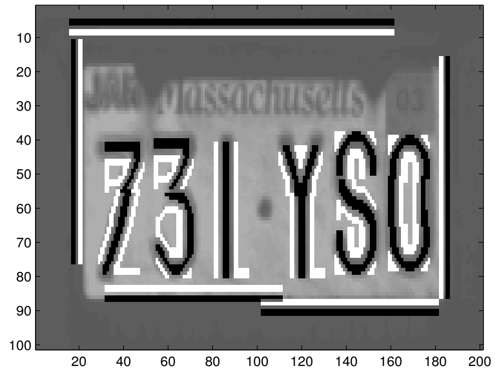


Figure 4.26: top string by level-2 filters

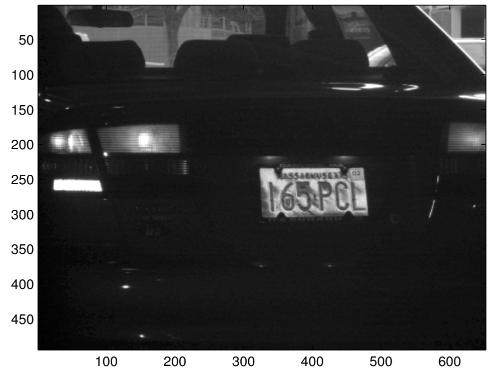


Figure 4.27: original image

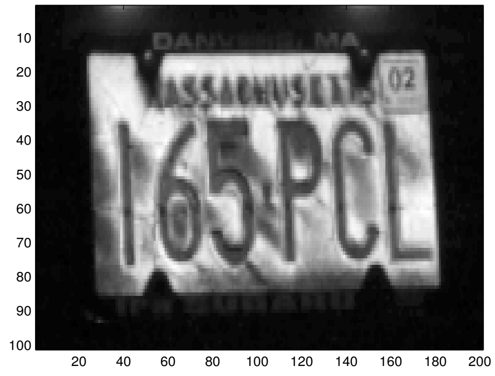


Figure 4.28: zoomed license region

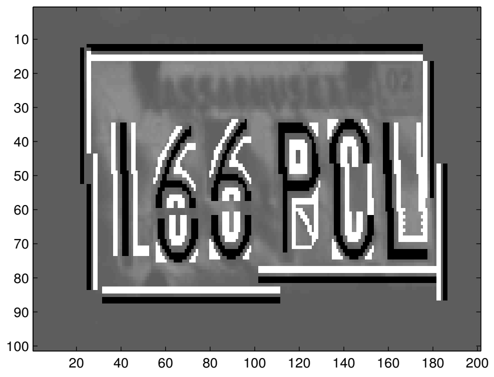


Figure 4.29: top string by level-2 filters

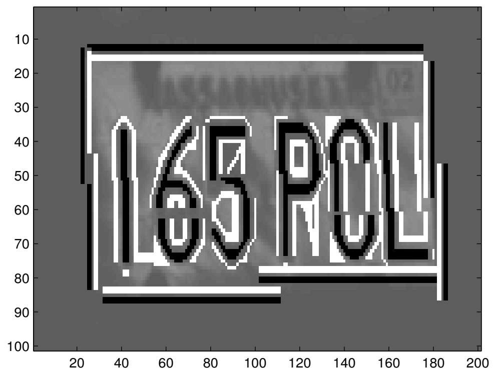


Figure 4.30: top string by level-3 filters

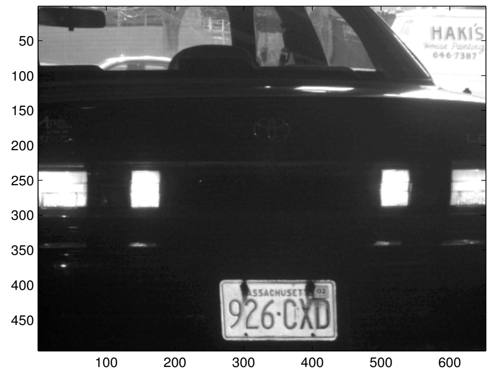


Figure 4.31: original image

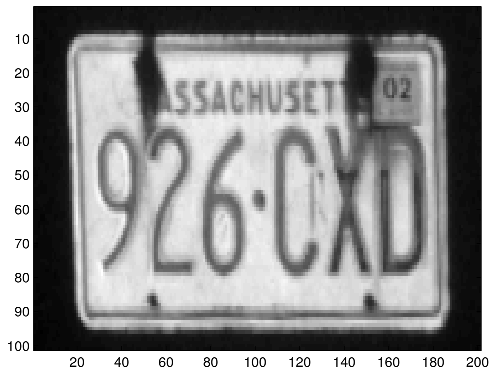


Figure 4.32: zoomed license region

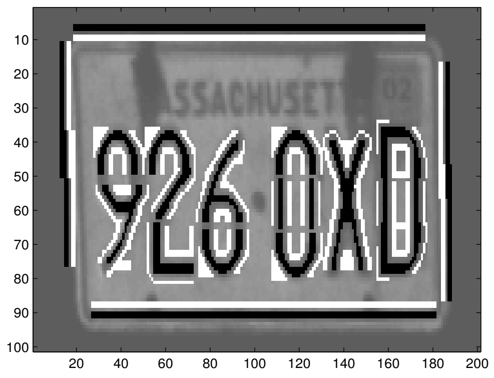


Figure 4.33: top string by level-3 filters

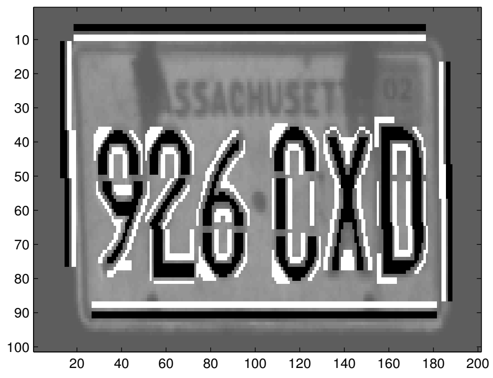


Figure 4.34: top string by level-4 filters

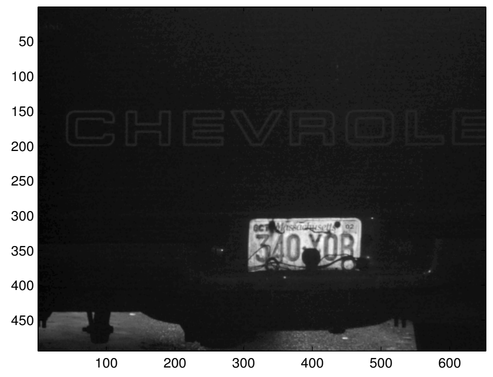


Figure 4.35: original image

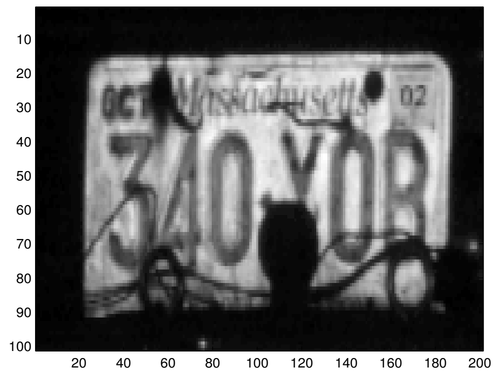


Figure 4.36: zoomed license region

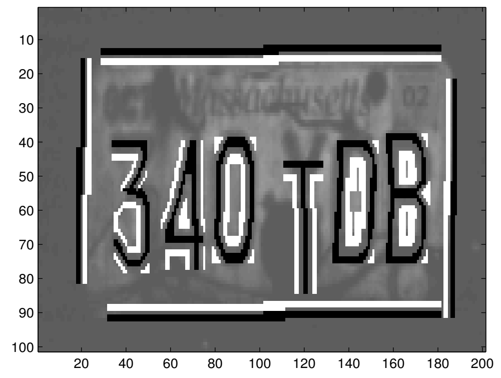


Figure 4.37: top string by level-1 filters

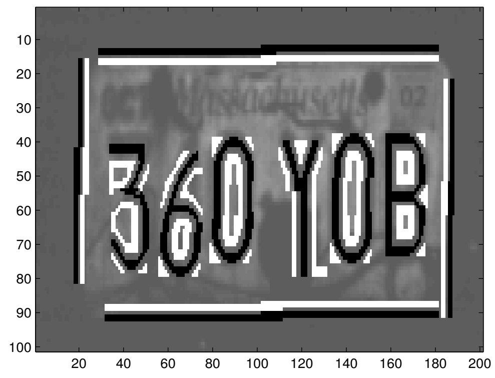


Figure 4.38: top string by level-2 filters

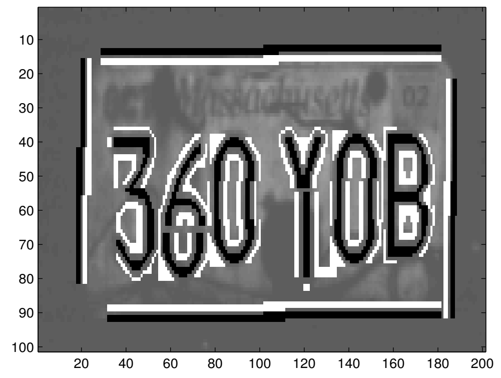


Figure 4.39: top string by level-3 filters

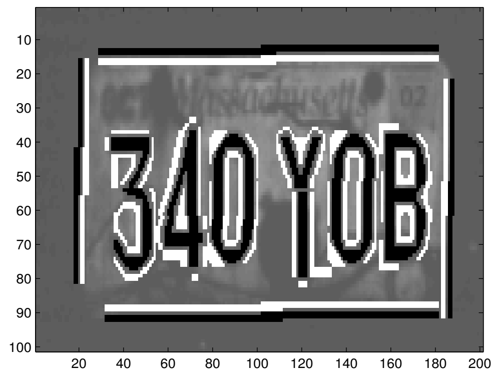


Figure 4.40: top string by level-4 filters

If we look at the license readings in Figures 4.37-4.40, we notice that the first one (using level-1 filters) contains two wrong characters, due to similarity of “Y” and “T”, “O” and “D”; the second one (using level-2 filters) corrects those two, but mistakes “4” as “6”; the third one shares the same reading as the second one; and the fouth one gets everything correct. A few things can be learned here. Usually you can get more noisy characters right with more discriminant filters. Although you sometimes get the same reading, more discriminant filters give you more precise positions. However, sometimes you may have new mistakes after you choose more discriminant filters because parts of the same level are competing with each other.

4.4.3 Relation versus Non-relation Parsing

Message: Relationship (non-Markovian term) matters in the parsing setting. It can alter the interpretation dramatically even if the data model is already very discriminative.

Brief description of the demonstration: We demonstrate this message through parsing a few test images under the non-relational setting (i.e., Markov backbone without a non-Markovian term) and under the relational setting (i.e., compositional distribution with the non-Markovian term) while keeping everything else unchanged. In either setting, the true objects are always in the candidate list, but a significant percentage of false positive candidate objects is removed at each layer under the relational setting. This will become even more obvious when less discriminative part filters are used.

Detailed description of the demonstration: Markov backbone is essentially a special case of the compositional distribution with its non-Markovian ratio being 1 (note: in general a non-Markovian ratio is a function of attributes). For a fixed image, we first activate terminal bricks which pass the thresholds; then list essentially all the objects that are compatible with the terminal bricks; next we equip each object with its likelihood ratio, i.e., $\frac{P(\vec{y}|X(I))P(X(I))}{P(\vec{y}|X(\phi))P(X(\phi))}$, under the Markov and non-Markovian settings (Note: $P(X(I))$ are different under the two settings); finally we conduct some comparisons under the two settings. Since the list of compatiable objects is substantial, we apply clustering and pruning for each layer as we did before.

This demonstration can be subdivided into two parts. The first demonstration is a comparison of the state configurations of the string-related bricks at each layer (from layer 2 and above) under the two settings. Fix a type of string-related brick, under the non-Markovian setting, remove all the poorly composed objects of this type (that is, an object with any involved non-Markovian ratio less than 1) from the above list, so we always get a subset of cadidates under the non-Markovian setting with respect to under the Markov setting. The point is that the non-Markovian term does play a selective role by keeping the true positive and throwing away only the false positives with a high percentage. The second demonstration is a comparison of the top string-related object (i.e., the string-related object with the biggest likelihood ratio) under the two settings, using filters with different discriminant levels.

Examples of the first demonstration are shown in Figures 4.41-4.57. Note: The color of an active grid (brick) represents the overall bits gain of the best candidate for that brick (with respect to the null). Inactive grids with a background color take value 0 and active grids with darker colors take negative values. Also, we only color the upper-left corner of an active brick's receptive field.

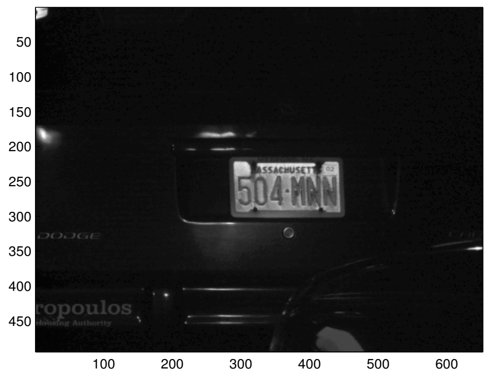


Figure 4.41: original image

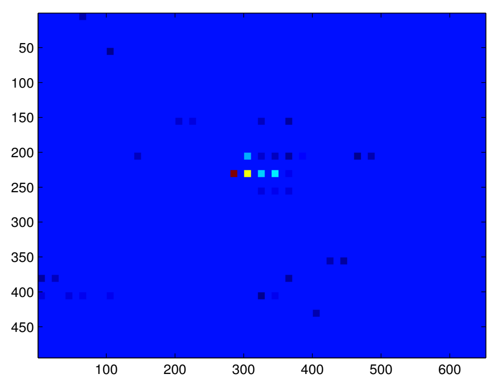


Figure 4.42: 35 active “5” bricks in no-relational setting

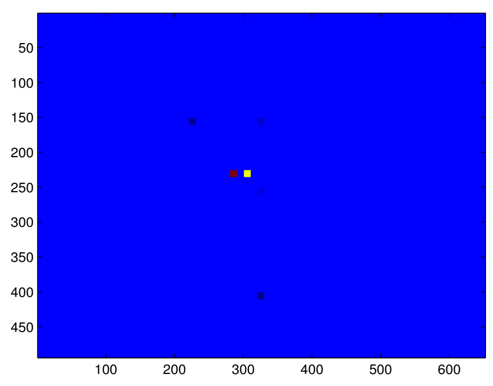


Figure 4.43: 6 active “5” bricks in relational setting

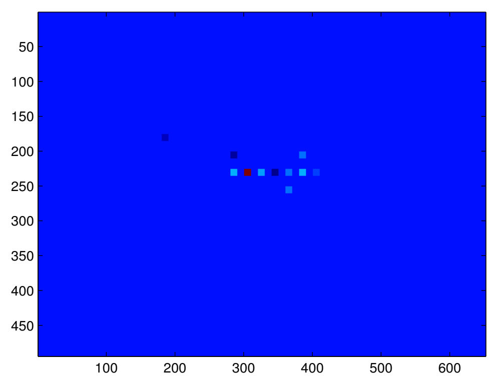


Figure 4.44: 11 active “0” bricks in no-relational setting

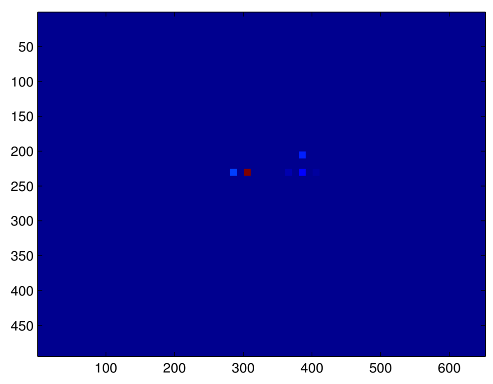


Figure 4.45: 7 active “0” bricks in relational setting

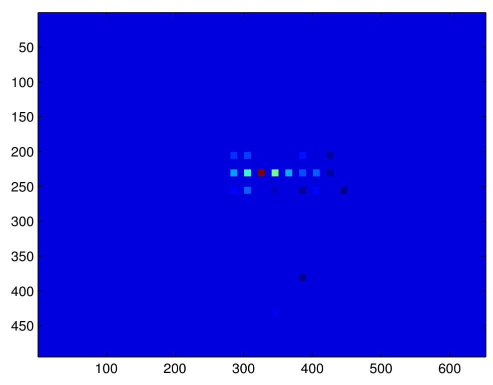


Figure 4.46: 21 active “4” bricks in no-relational setting

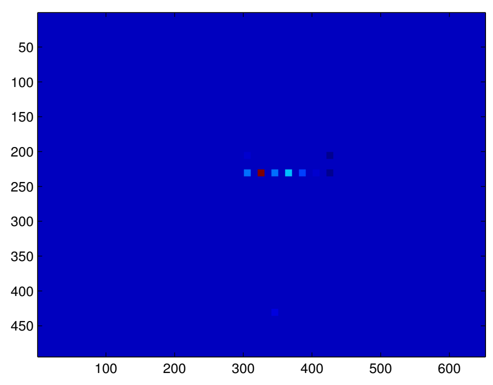


Figure 4.47: 13 active “4” bricks in relational setting

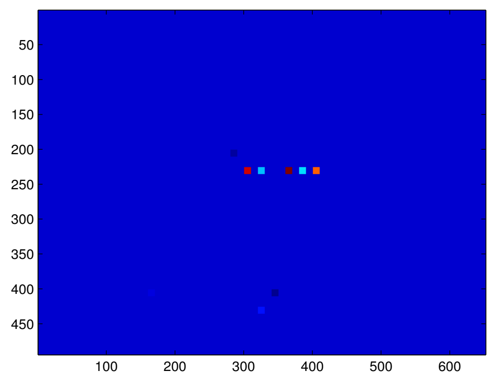


Figure 4.48: 11 active “M” bricks in no-relational setting

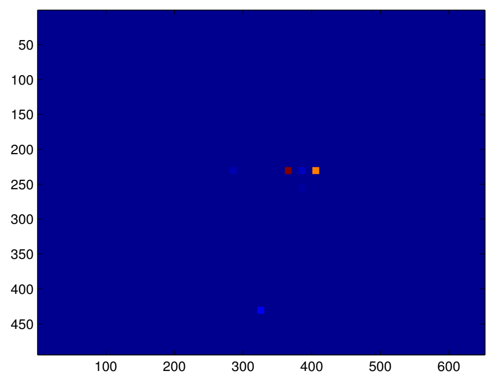


Figure 4.49: 6 active “M” bricks in relational setting

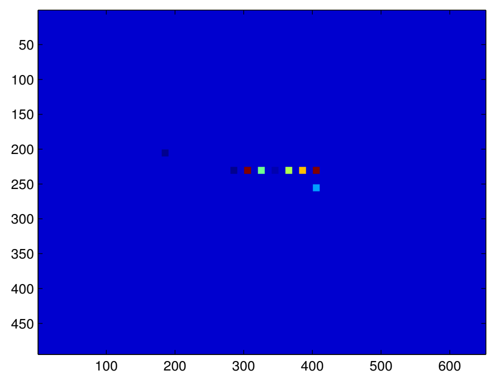


Figure 4.50: 11 active “N” bricks in no-relational setting

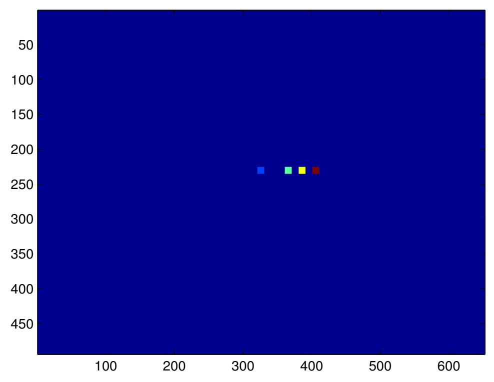


Figure 4.51: 5 active “N” bricks in relational setting

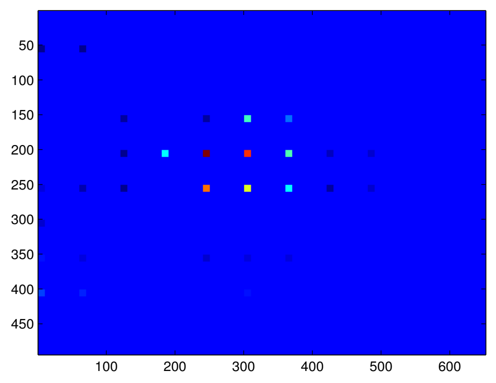


Figure 4.52: 31 active 3-digit bricks in no-relatonal setting

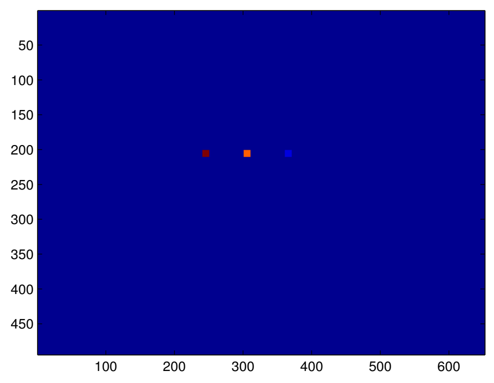


Figure 4.53: 3 active 3-digit bricks in relational setting

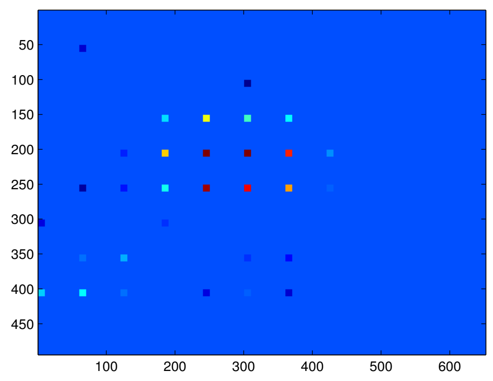


Figure 4.54: 31 active 3-letter bricks in no-relational setting

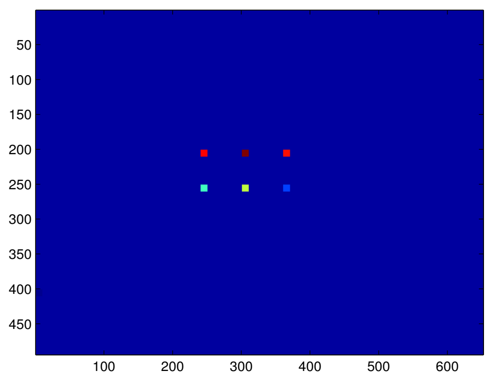


Figure 4.55: 7 active 3-letter bricks in relational setting

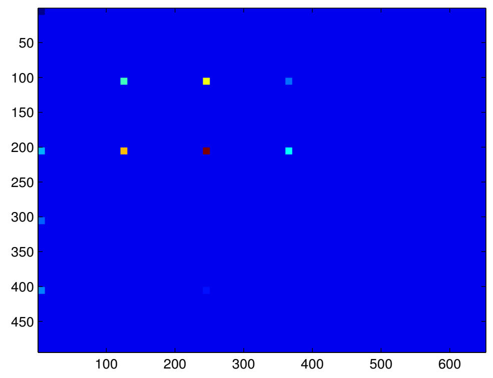


Figure 4.56: 12 active string bricks in no-relational setting

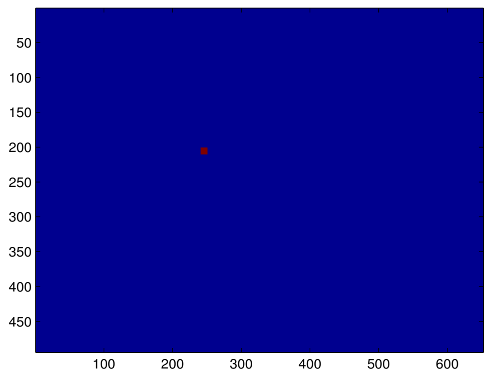


Figure 4.57: 1 active string brick in relational setting

To conserve space, the pictures above are only for the characters and partial strings in the actual license reading. The same effect occurs for any other character and partial string.

In the second demonstration, we compare the top string-related objects under the two settings for level 1, 2, 3 and 4 set of filters respectively. For less discriminat filters, it is quite time-consuming to generate the list of all compatiable string-related objects out of the whole image. We relax the condition to get around the computational issue while making the argument even stronger. That is, we make use of the non-Markovian property of a license boundary to detect its position first (assume we always get the right one), then only the string detection process is applied within the boundary. Not surprisingly, even over such a small patch, a variety of mistakes could occur under the no-relational setting. In fact, the top string-related objects out of the whole image and out of the subimage are the same in most cases. This is because the most structured region, the license plate region, is often surrounded by a large homogeneous region, and no widely-spreaded object can exist due to the binding effect of the Markovian structure. If the true reading is not the best string-related object in the subimage of the license region, it could not be the best string-related object in the whole image either. Examples of the second demonstration are given in Figures 4.58-4.85.

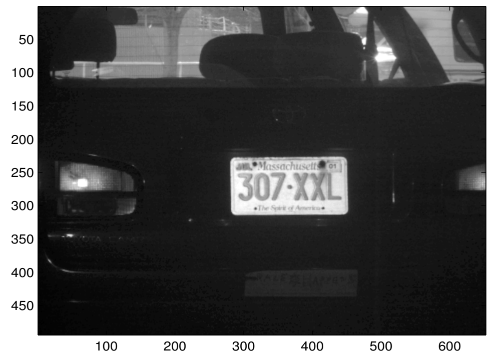


Figure 4.58: original image

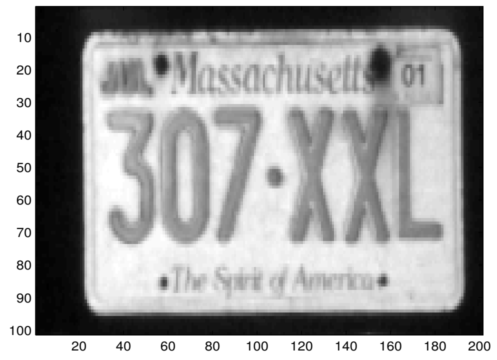


Figure 4.59: zoomed license region

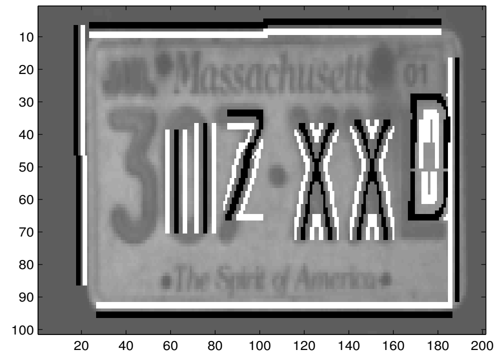


Figure 4.60: top string under the no-relational setting using level-1 filters

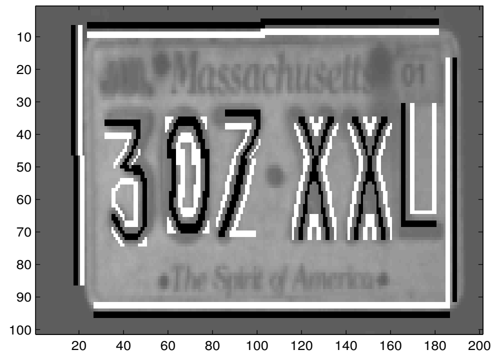


Figure 4.61: top string under the relational setting using level-1 filters

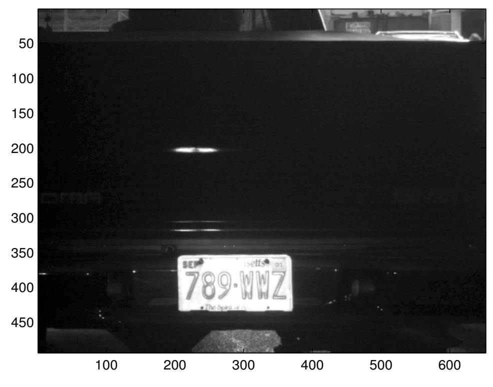


Figure 4.62: original image

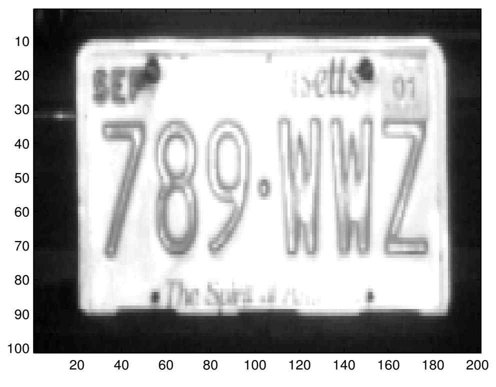


Figure 4.63: zoomed license region

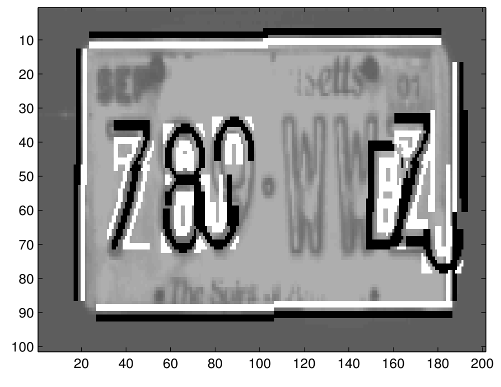


Figure 4.64: top string under the no-relational setting using level-2 filters

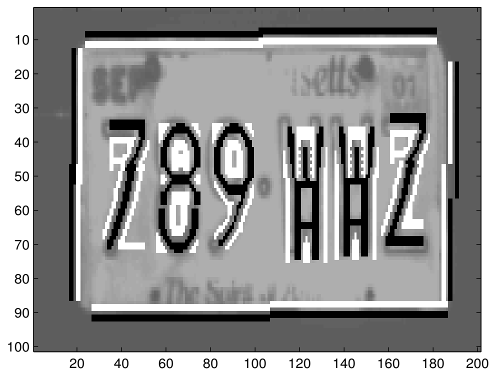


Figure 4.65: top string under the relational setting using level-2 filters

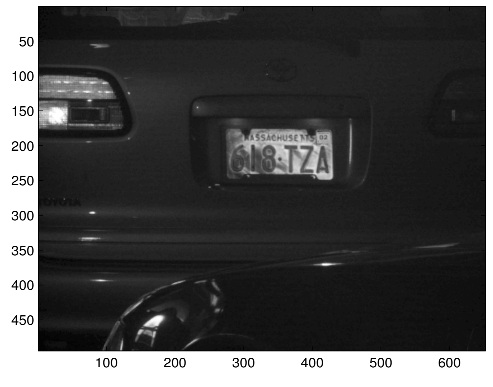


Figure 4.66: original image

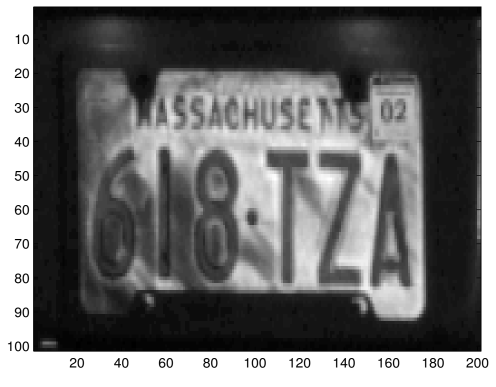


Figure 4.67: zoomed license region

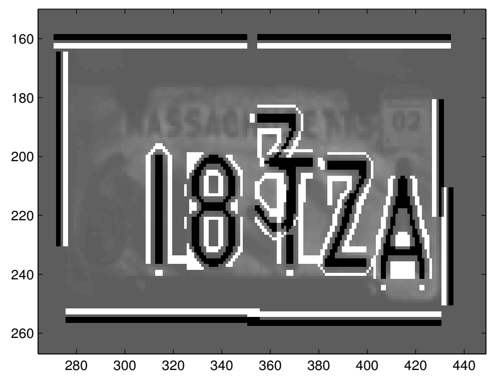


Figure 4.68: top string under the no-relational setting using level-3 filters

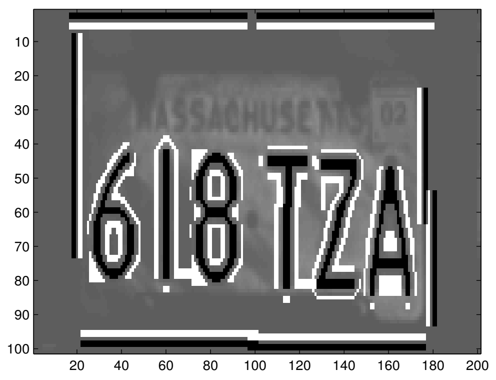


Figure 4.69: top string under the relational setting using level-3 filters

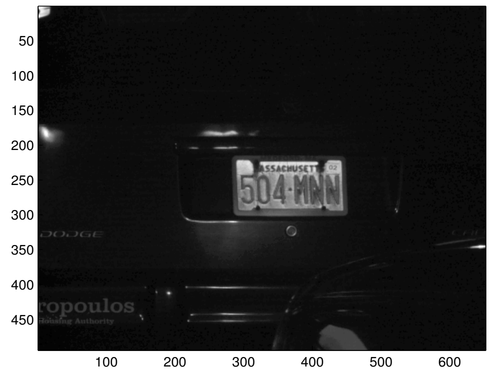


Figure 4.70: original image

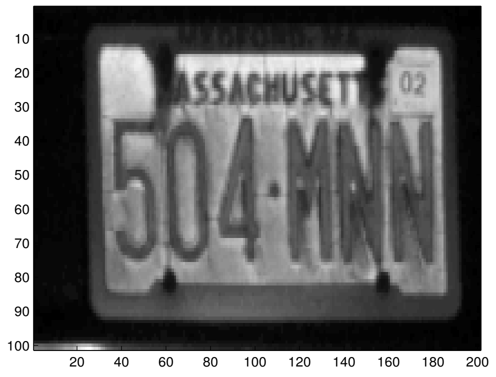


Figure 4.71: zoomed license region

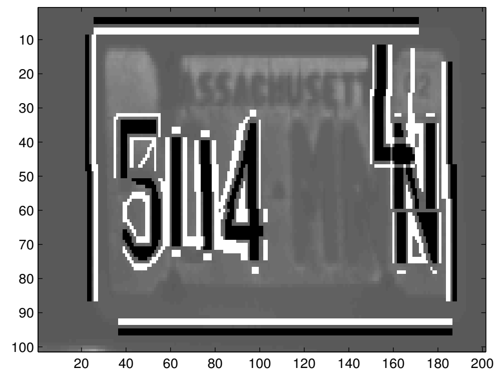


Figure 4.72: top string under the no-relational setting using level-4 filters

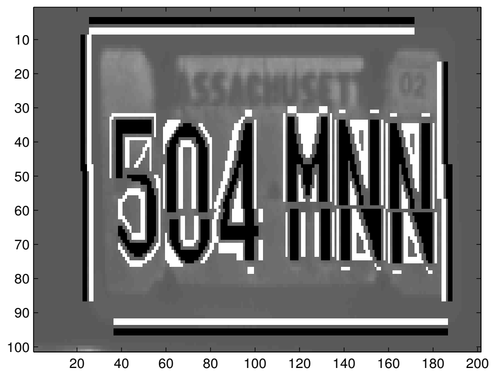


Figure 4.73: top string under the relational setting using level-4 filters

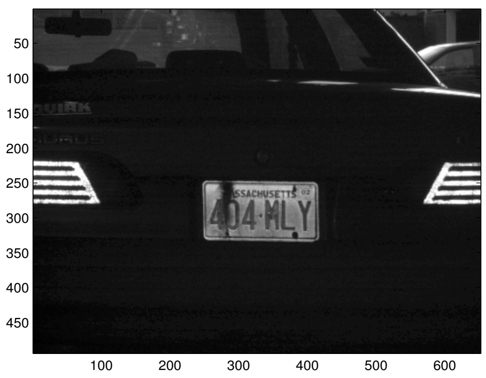


Figure 4.74: original image

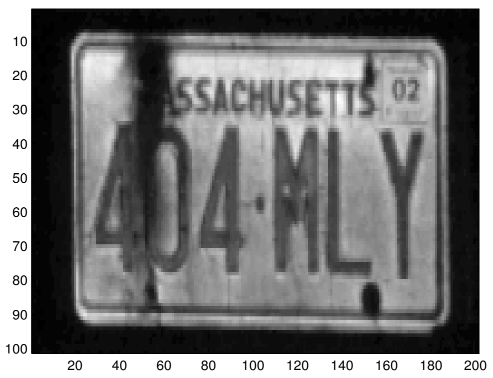


Figure 4.75: zoomed license region

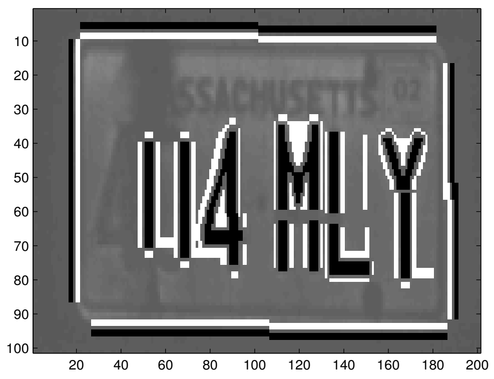


Figure 4.76: top string under the no-relational setting using level-4 filters

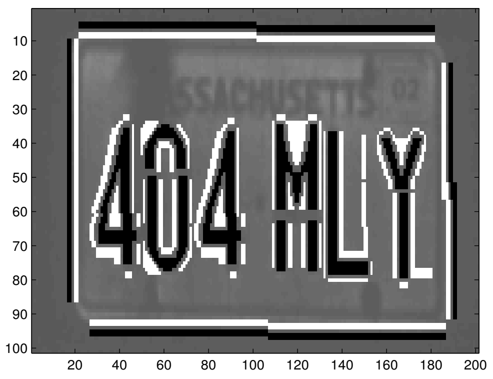


Figure 4.77: top string under the relational setting using level-4 filters

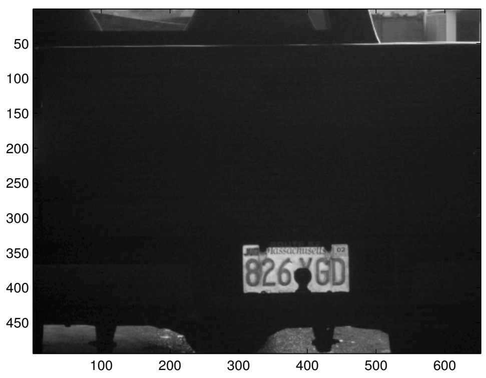


Figure 4.78: original image

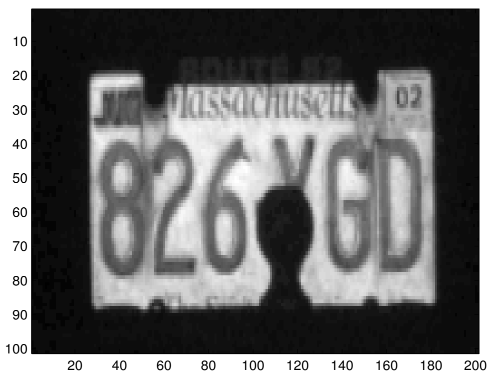


Figure 4.79: zoomed license region

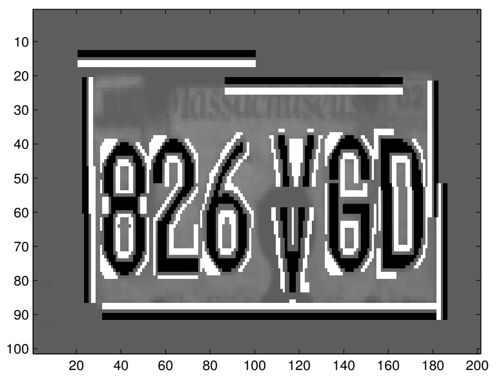


Figure 4.80: top string under the no-relational setting using level-4 filters

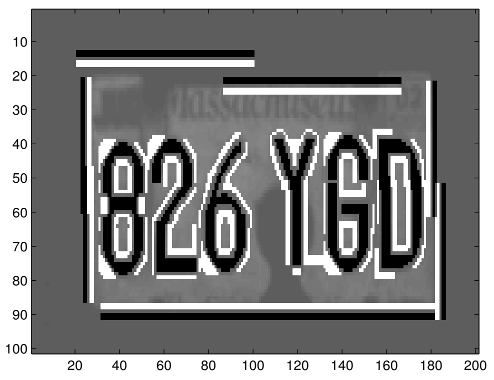


Figure 4.81: top string under the relational setting using level-4 filters

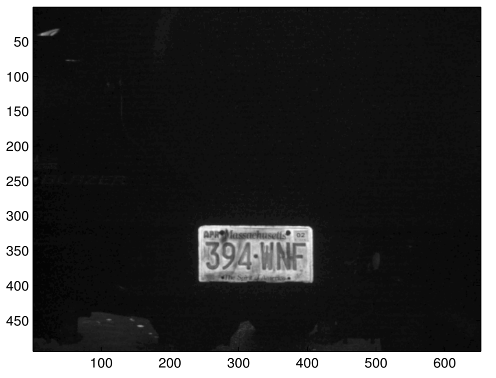


Figure 4.82: original image

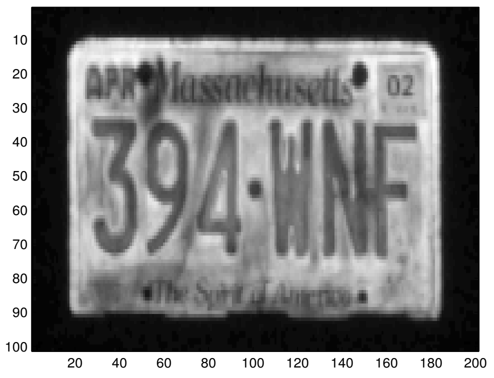


Figure 4.83: zoomed license region

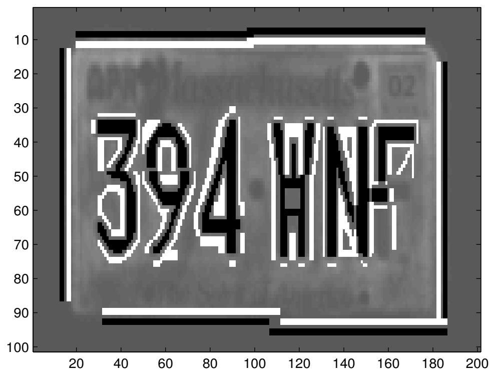


Figure 4.84: top string under the no-relational setting using level-4 filters

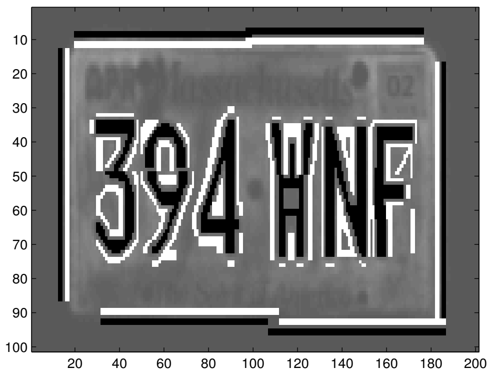


Figure 4.85: top string under the relational setting using level-4 filters

From the preceding pictures, we can see that relationship does play an important role, no matter which level filters are applied. The output reading out of the no-relational setting can be vastly different (see Figure 4.72) or minutely different (see Figure 4.80) from the true one, or even the same, but sometimes with imprecise positions of its parts (see Figure 4.84). The reason for such differences are straightforward. For example, string "5114 LN" in Figure 4.72 is the top string-related object under the Markov backbone, due to its good data likelihood ratio and no penalty for the poor composition between its partial strings; the true string "504 MNN" in Figure 4.73 is the top one under the non-Markovian setting, due to its large non-Markovian term.

4.4.4 False Positives under Whole Model versus Parts Model

Message: Fix a detection rate, the number of false positives are reduced under the parts model, compared to that under the whole model. This is true in a very general setting. (see the main theorem in §3.3.2)

Brief description of the demonstration: Consider a compositional machine with only two layers; we demonstrate this message through comparing the number of false positive characters of a fixed type in a fixed test image under the two models. A significant percentage of false positives is removed under the parts model.

Detailed description of the demonstration: There are quite a few ways to demonstrate this message. Two of them are described and demonstrated with figures.

Method 1: Fix a character in a test image, first apply the list generation process to the terminal layer without setting any threshold; then set the threshold of bits gain for the character to be 0 (note: positive bits gain implies that data is better interpreted as the character than nothing), and apply the list generation process for each brick of that character on the second layer independently (note: a character brick may have multiple candidates); finally we compare the best candidate at each brick (i.e., the candidate with the biggest bits gain from which pass the threshold) with the bits gain of its sub parts (for a 2-part character) and possible combinations (for a 3-part character), and we only keep the bigger one under the parts model. That is, if the whole character has fewer bits gain than its subpart(s), we remove it (set the brick off) under the parts model. Under the whole model, if any candidate of that character passes the threshold (0), its corresponding brick is set to be on.

So we have two maps of active brick configurations of that character type, the one under the parts model is always a subset of the one under the whole model. Usually a high percentage of false positive characters is removed under the parts model, while the true positive character stays there. Some demonstration pictures are listed in Figures 4.86-4.96. The same color convention (mentioned in §4.4.3) is applied here.

Note: Method 1 conducts a very conservative comparison, i.e., comparing the best character instantiation with its own part. A less conservative one could be a comparison between the best character instantiation with the best part in the RF of that character brick. In that case, we would expect a higher percentage of false positive characters removed. Again, here we only show the relevant characters to save some space, and the same thing happens for any other character. Also, we use the same σ in p^c as the one used in general detection, which is quite loose. Strict binding (small σ) could further increase the number of false positives removed.

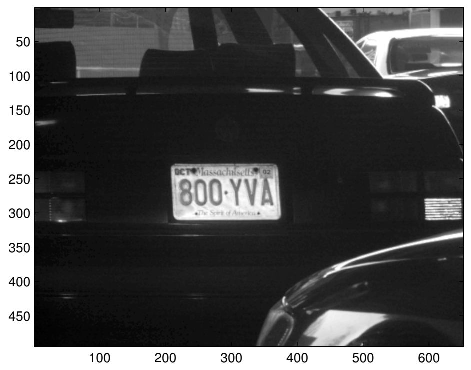


Figure 4.86: test image

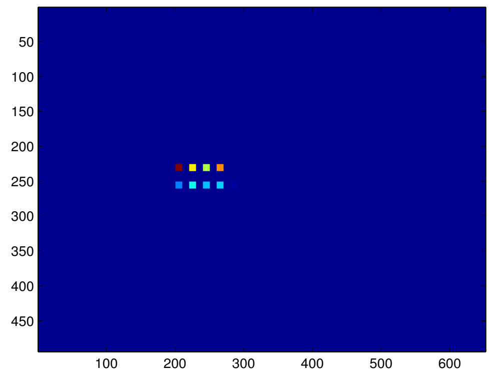


Figure 4.87: 9 active "8" bricks under whole model

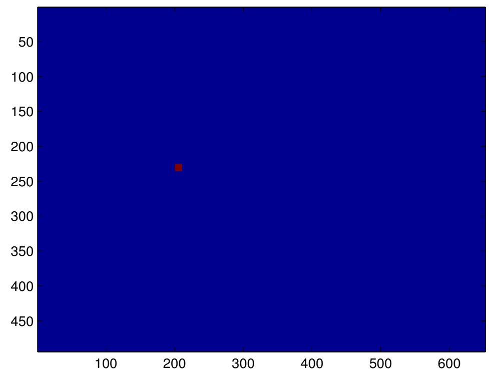


Figure 4.88: 1 active "8" bricks under parts model

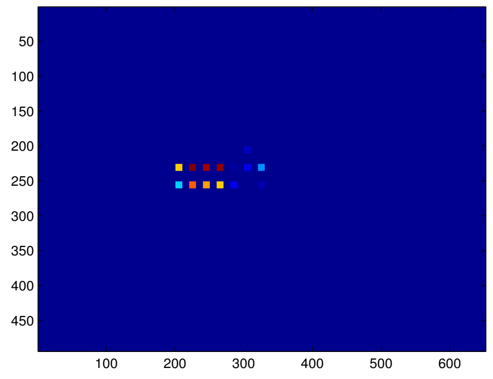


Figure 4.89: 14 active "0" bricks under whole model

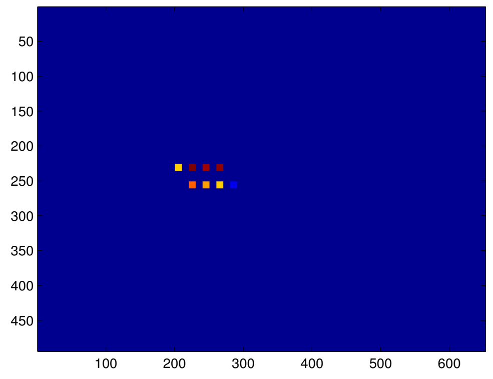


Figure 4.90: 8 active "0" bricks under parts model

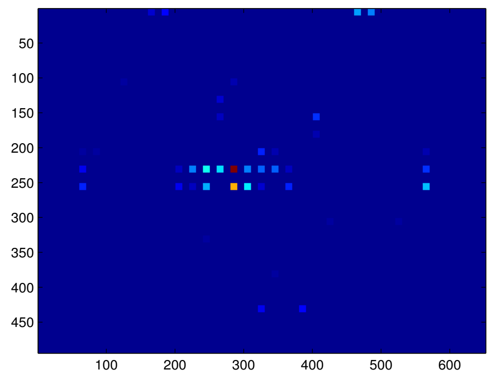


Figure 4.91: 53 active "Y" bricks under whole model

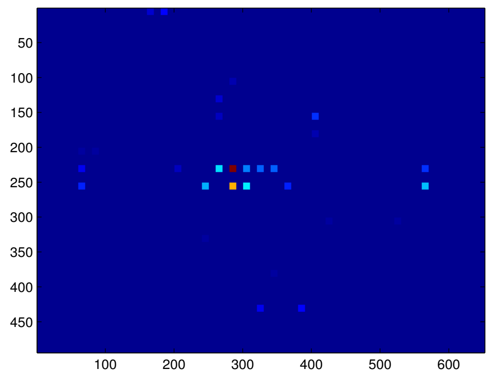


Figure 4.92: 38 active "Y" bricks under parts model

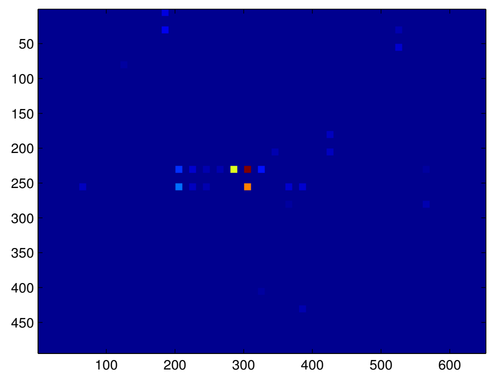


Figure 4.93: 33 active “V” bricks under whole model

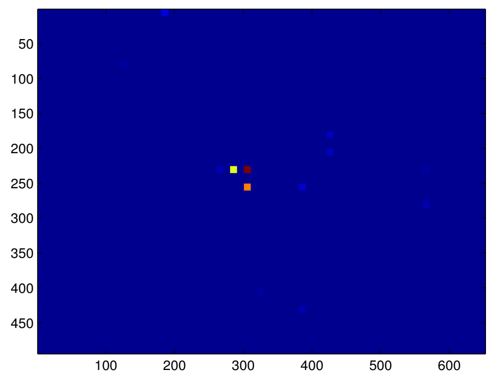


Figure 4.94: 17 active “V” bricks under parts model

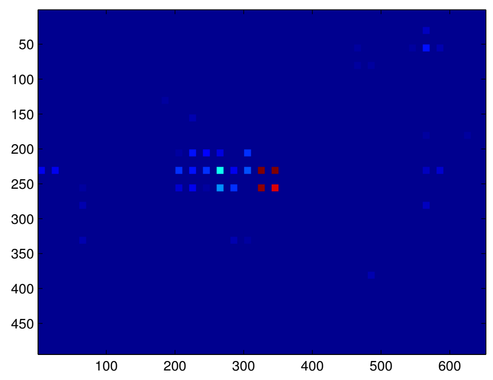


Figure 4.95: 53 active “A” bricks under whole model

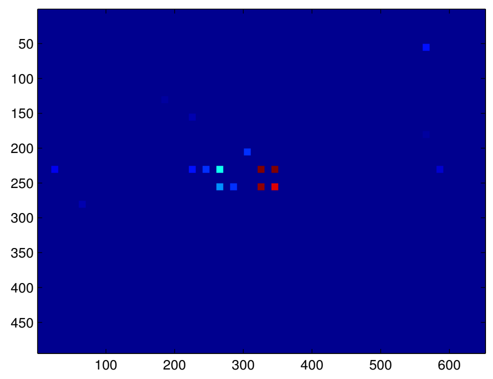


Figure 4.96: 20 active “A” bricks under parts model

Method 2: Fix a character in a test image, under the whole model, we apply the list generation process to the terminal layer without setting any threshold and apply the list generation process to the character layer with a sequence of thresholds; under the parts model, we apply the list generation process to the first two layers using carefully chosen threshold for each layer, which are derived analytically to maintain a fixed power of the sequential testing. This method is best suited to a demonstration of a ROC curve, which can be achieved by applying a sequence of powers.

For a fixed power β, we derive the threshold for each part as follows:

$$P(\frac{P(data|part)P(part)}{P(data|null)P(null)} \geq c'(\beta)|part) = \beta$$

Since the rank sum statistic is sufficient in the data likelihood ratio, and it obeys exponential law when the part brick is active and normal law when the part brick is off, the above form can be simplified to:

$$P(f(R) > c'(\beta)) = \beta$$

where f is a known functional form (ratio of two distributions) and R is the rank sum statistic, exponentially distributed.

For a fixed power β, we derive the threshold for a character as follows:

$$P(\frac{P(data|character)P(character)}{P(data|parts)P(parts)} \geq c(\beta)|character) = \beta$$

By the independence assumption (i.e., given the configuration of terminal layer, data is independent of higher layers), the data likelihood ratio is cancelled, and the above form can be simplified to:

$$P(g(d_X, d_Y) > c(\beta)) = \beta$$

where g is a known functional form (i.e., the likelihood ratio of composition versus non-composition in §4.1.1.2) and d_X, d_Y are relative x-axis distance and y-axis distance between composing parts, which are independently and normally distributed.

We compute the cumulative distribution functions for f and g above, and by inversing them, a sequence of $c(\beta)$ and $c'(\beta)$ can be read out from a sequence of fixed β. These theoretical thresholds give us guidance towards picking proper thresholds for a character's parts and their composing relationship. Under the parts model, a character is claimed to be present if and only if each of its parts passes the test and the relationship among them passes the test.

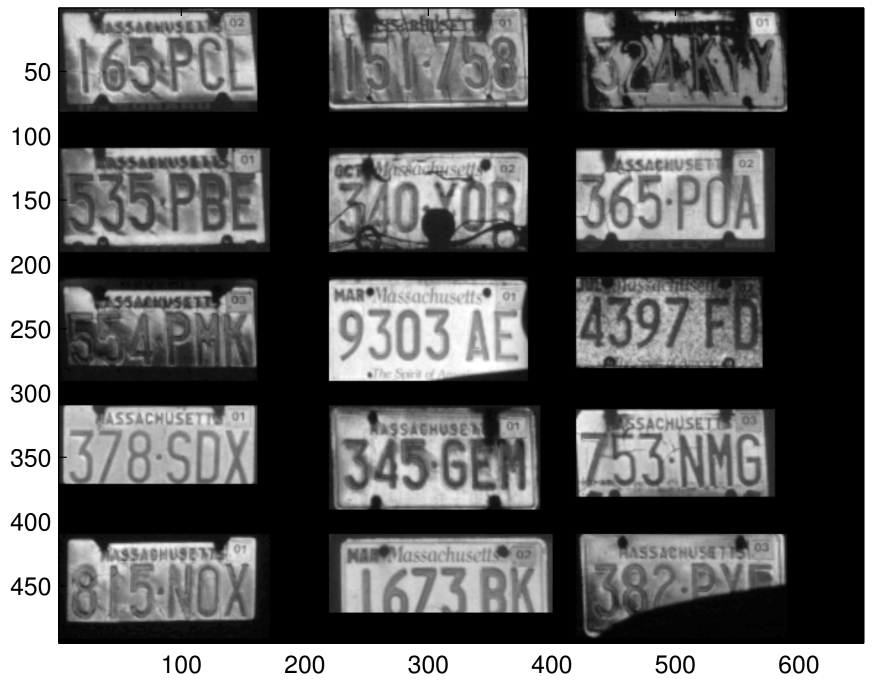
Given that we have a sequence of thresholds under both models, we should be able to generate two ROC curves. Due to several factors (lack of data, discriminant part filters, threshold sensitivity, etc.), the curves are very singular, and we only use this method for a point-wise comparison. That is, for a fixed detection rate, we compare the number of false positives under the two models. The demonstration is done on a test image full of plates with added Gaussian noise. The convention of counting the true/false positives is defined as follows.

Given that we know the true positions of a fixed character (say "3") in a test image,

1. If any candidate character ("3") under whole/parts model is within a certain radius of one of the true characters ("3"), then one more true positive is counted. If more than one candidate ("3") is within that radius, we only count it once.

2. If any candidate character ("3") under whole/parts model is outside the radius of any of the true characters ("3") and any of the current false positive characters ("3"), then one more false positive is counted. When more false positives ("3") fall into the radius of one of the current false positives ("3"), they are not counted.

In the demonstration images below, we compare the number of false positive characters ("3") under the two models, given the perfect detection rate. To be more precise, the numbers of false positives quoted for comparison under two models are the best (smallest) we can achieve while not missing any of the true positives. The radius for count convention is set to be 5. We color a correctly-detected region (upper-left corner of "3") white and a wrongly-detected region dark.



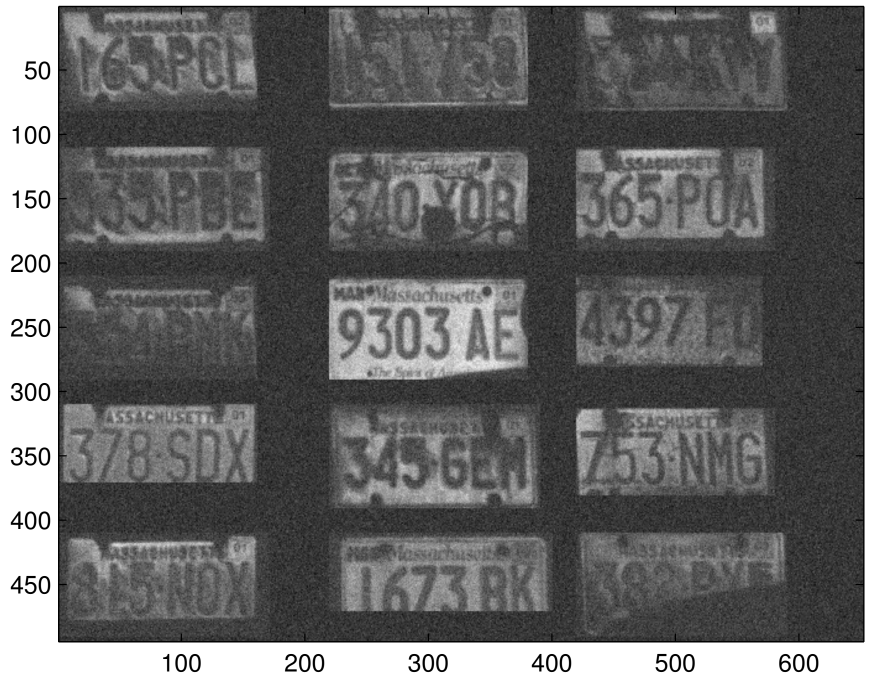
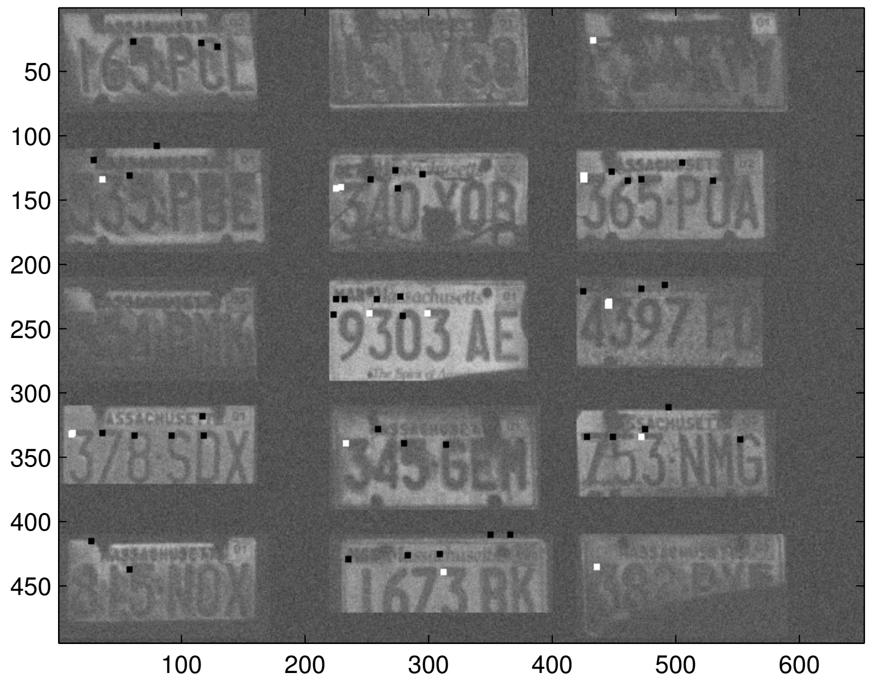


Figure 4.97: test image and its noisy version with additive N(0,20) noise



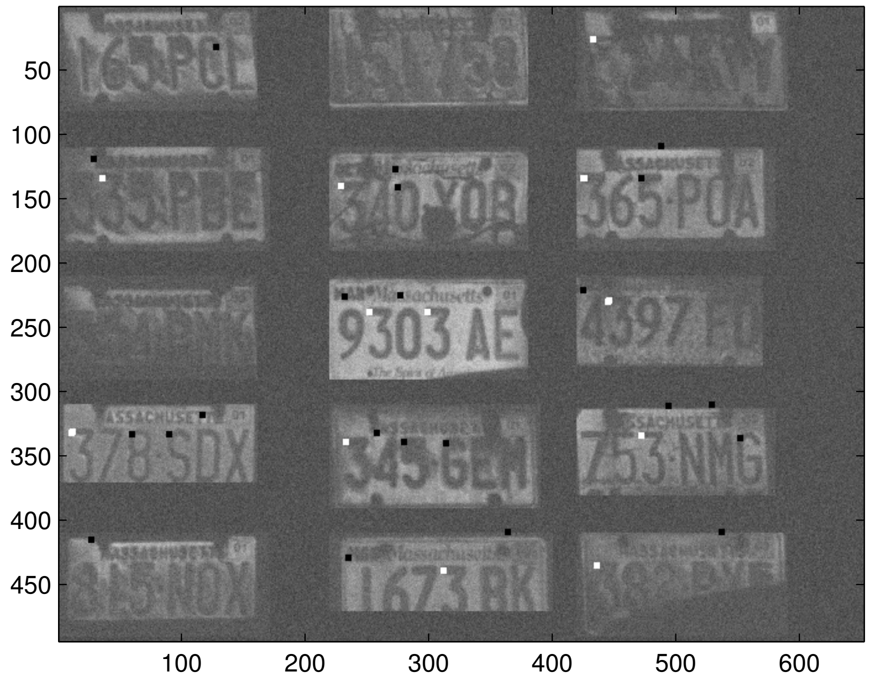


Figure 4.98: 44 false positives under the whole model (above) and 22 false positives under the parts model (below), with 12 true "3" all detected.

An interesting note: under the whole model, even if we allow two targets ("3") to be missed, we still have 41 false positives, which is larger than the 22 false positives with perfect detection under the parts model.

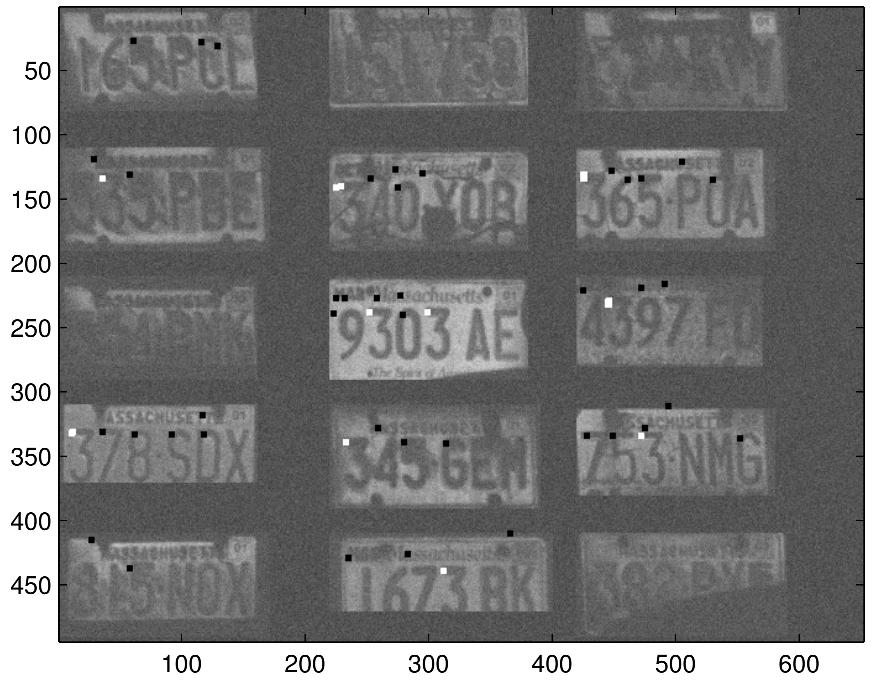


Figure 4.99: 41 false positives under the whole model with 10 out of 12 true "3" detected.

The above two methods are closely related. The first one is more like a bits-gain driven scheme to parse a scene and the second one is more like a sequential-test scheme to test a scene. Assuming a good relationship among parts, if a character has fewer bits gain than one of its parts (in method 1), it usually implies that its other part is too poor to pass the threshold (in method 2) and vice versa; if we assume a poor relationship among parts, it leads to fewer bits gain of a character compared to either of its parts (in method 1), which is equivalent to failing the test of relationship (in method 2).

We demonstrate this message only through the character layer, due to simplicity and computational feasibility. For the higher layers, the receptive field of each brick grows wider, consistency checking needs to be introduced (not necessarily for the character layer) and candidate generation under the whole model is too time-consuming.

4.4.5 General Detection in Cluttered Background

Message: The construction of a composition system that coherently implements a hierarchy of contextual constraints leads to great performance.

Brief description of the demonstration: We demonstrate this message through a set of test images and a few synthesized images. The test is conducted over 385 images picked from approximately 400 images with a Massachusetts license plate in them (the deleted ones contain license readings which are either inconsistent with our defined composition rules for a string, i.e., 4-digit + 2-letter, 3-digit + 3-letter or a significent percentage is truncated) and a few synthesized images with multiple license plates. The code can deal with an uncertain number of plates (0 or more) in one image as long as they don't overlap to a large degree. The detection rate is about 99.5% at the character level and is above 98% at the plate level. The false positive number at the plate level is 0.

Detailed description of the demonstration: Fix an image and fix the threshold for the rank sum statistic of each terminal part to be its lower 30 percent (lower rank sum implies better match, 50 percent corresponds to the mean under the null), all the parts with the lower rank sum enter the list and corresponding terminal bricks are activated. Higher layer thresholds for the non-Markovian ratios are set to guarantee good binding relationship. List generation process, clustering and pruning for higher layers are applied subsequently. Given the generated list, the pseudo list optimization scheme selects K (or whatever number when it stops) consistent objects from it to form a parse during each run. The final parse for the image is the best parse (with biggest bits gain) among several runs. For the purpose of parsing an entire image, string-related and frame-related bricks are both visited. The running time depends on K (i.e., the detail level of the parse).

Current Speed-up Scheme

For the application of license detection only, we have a speedup version. Three major steps are involved.

1. Find the boundary of each potential license plate.

Since the license boundary has a similar hierarchical structure to the string, similar processes are applied. That is, we start with computing the rank sum statistic for each of the four boundary terminal parts; for each part (passing its threshold), we activate its corresponding brick. Similar list generation process, clustering and pruning are applied for the higher layers in the boundary hierarchy. Finally we apply pseudo list optimization to candidate boundary objects only to select consistent ones from the list until there is no more bits gain. Assuming one license plate will not be occluded by another one to a great degree, we cluster the chosen boundaries by their positions. As the binding relation for a boundary is not very strict, we may have false positive boundaries.

2. Given the position of each potential license boundary, find the reading (if there is one) within the boundary, that is, a subimage with size 100×200.

Now we are only visiting a scaled-down version of string hierarchical structure. That is, given the subimage, we compute the rank sum matrices for terminal parts of the characters, conduct relevant tests and activate certain bricks. Similar list generation process, clustering and pruning are applied for the higher layers in the string hierarchy. Finally we pick the best consistent string (if there is one) as the reading for the potential license plate. The binding relation for a string is quite strict; it is rare to have false positive strings.

3. We then put all the candidate strings generated in step 2 into a list, apply pseudo list optimization, and the output string(s) will be the final reading(s). At this stage, no false positive strings ever exist.

Extension of Current Speed-up Scheme

There is an even more general search strategy, aimed at detecting license plates with minimal cost. It is a depth-first search strategy over the same hierarchical structure. Briefly speaking, for each license brick on the top layer, we conduct a CTF search (testing) over its receptive field to decide whether a license object is present. The most appealing part of this strategy is that it spends the most time only on the places where it is most needed, which is quite different from the current bottom-up search strategy (spending time equally all over the image). A detailed description of this search strategy is as follows.

Fix a top-layer brick, pick one of its active states, which determines one spatial configuration of its children bricks. Then, pick a child brick and one of its active states, which determines one spatial configuration of its children bricks. Continue this process until the chosen child brick reaches the leaf of the hierarchical structure (i.e., terminal layer). Next, we collect the rank sum statistics over the small image patch that the terminal brick covers and conduct a simple test to decide whether the part is present (and the precise position) or not. If it is not present, we don't bother to check other terminal bricks which are in the same children set as this one, we just pick another active state for the parent brick (of this terminal brick) and do the similar depth-first search; if the part is present, we need to check other terminal bricks in the same children set and a possible composition among them if they all pass the tests (just as we did in the bottom-up scheme). In either case, we store whatever information we have achieved (say rank sum statistics, status of test, positions, etc.) for reusable purpose. For a brick on an intermediate layer, we may have multiple candidates and we have to store all of them (in a similar structure as in bottom-up scheme). Similar pruning and consistentcy check have to be done during the search.

The search ends after we loop through all the top bricks.

The key element in this depth-first search strategy is picking the right order. In the above description, we mentioned "pick one of its active states, pick a child brick," but the order of picking them matters in terms of the running cost. Intuitively, we want to search things first which require the least amount of effort so that we can narrow the search space as quickly as possible. Formally, the searching cost of an entity involves the number of its instantiations and the cost for each instantiation.

Consider the worst case scenario, i.e., every instantiation is possible. We need to enumerate all the instantiations for each brick and select an order among them. If we assume the cost for each instantiation is linearly related to the number of involved terminal parts, then a good heuristics (for picking the order) is the sum of all active states of all involved terminal parts, given that entity brick is active. States of reused parts are counted only once, since we stored the information after the first visit.

Consider the best case scenario, i.e every instantiation is impossible. For each instantiation, we only need to test one child of each children set at each layer, because it always fails. In this case, a good heuristics is the sum of all active states of the terminal parts which contain the interest point (given only one interest point per object). States of reused parts are also counted only once for the same reason.

The ideal heuristics is a mixture of the two, combined with the confidence (probability) in either case. As it is not hard to imagine, at the beginning of the search, the best case often occurs and at the end, the worse case that could happen.

We apply the above heuristics to pick the order of the depth-first search for our license detection application. Exact computation of the heuristics can be difficult, due to the reusability. Instead, we compute the approximations of them by ignoring the fact that parts are reusable, meaning the same part with the same state can be tested multiple times (equivalent to not storing the information after the first visit). The approximation is always bigger than the corresponding heuristics and the gap between them depends on the extent of reusability involved. Note: the goal here is not about estimation of an absolute testing cost for each brick, but about finding an optimal relative testing order. An example of approximated heuristics for a general digit is given below. The color of each grid represents the number of times the grid has been visited.

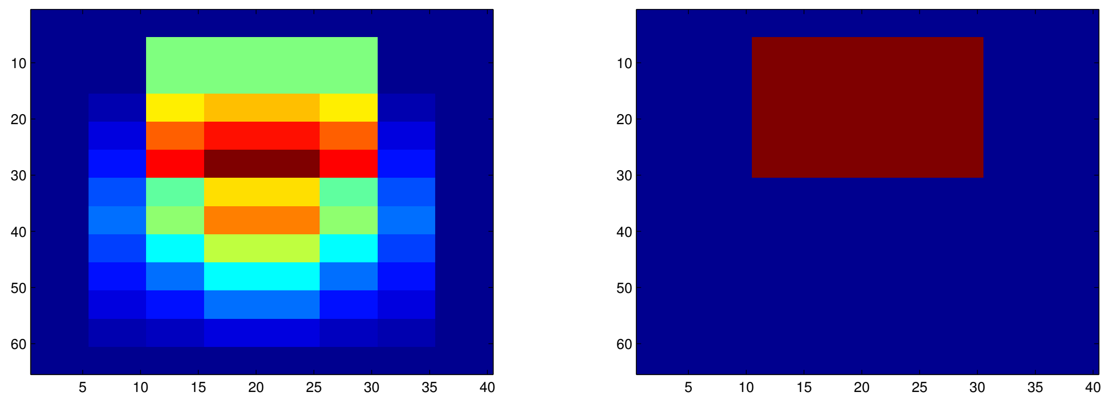


Figure 4.100: its receptive field with overlap (worst case) and its interest points region with overlap (best case)

The approximated heuristics for the worst/best case is equivalent to the sum of the above left/right region, with some positions being counted more than once. They are 153000 and 5000 for worst case and best case respectively.

The criteria we apply to pick the order is to compare the approximated best-case heuristics (the smaller one comes first). When we meet a tie, the one with smaller approximated worst-case heuristics comes first. So not surprisingly, fix a license brick and an active state, (child) frame brick is chosen before (child) string brick, due to the fact that frame brick has a much smaller sum of areas covered by possible interest points. This is very consistent with the bottom-up speedup version, which finds the frame first to narrow the search (to a subimage) and then looks for string (inside the subimage). Similarly we have a order for other "competing" entities, for example, 3-digit is checked before 3-letter, 2-letter is checked before 4-digit.

A further possible improvement can be made by introducing smaller terminal parts, because they are heavily reusable and testing on them can be very fast (not to mention that they can deal with deformable templates).

Note: All the demonstration pictures below are generated from the bottom-up scheme (or its speed-up version, not from the general depth-first search), since we care more things (such as scene parsing) than just license detection.

4.4.5.1 Demonstration of License Detection

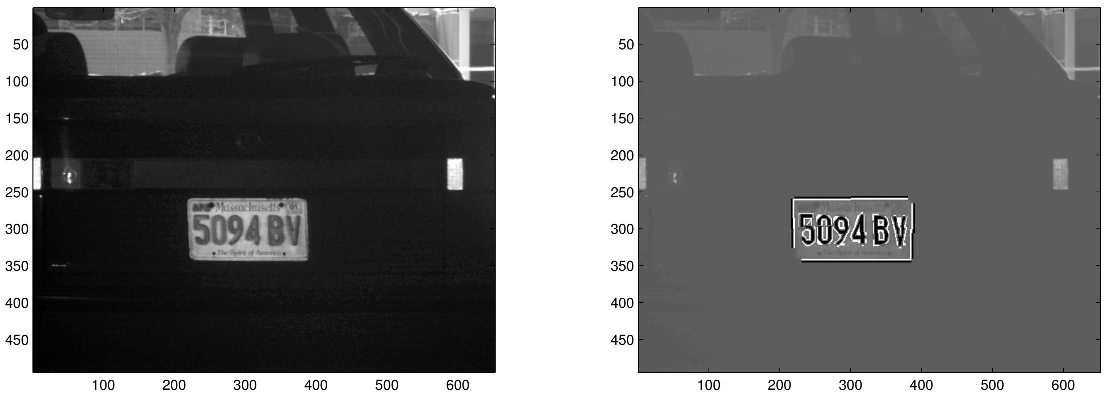


Figure 4.101: Original image and its top object

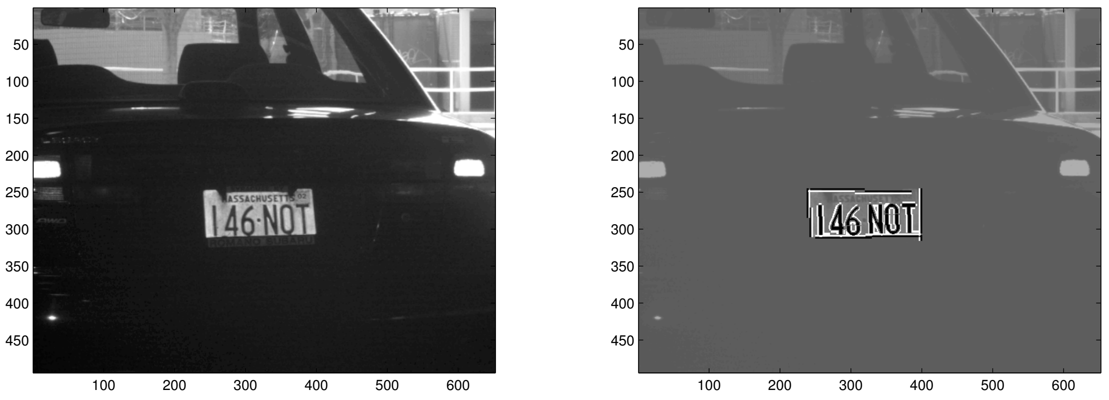


Figure 4.102: Original image and its top object

Note: Small changes in the scale of characters, thickness of strokes and viewing angles are allowed. For big changes, more states for relevant bricks or more bricks have to be introduced to cover them.

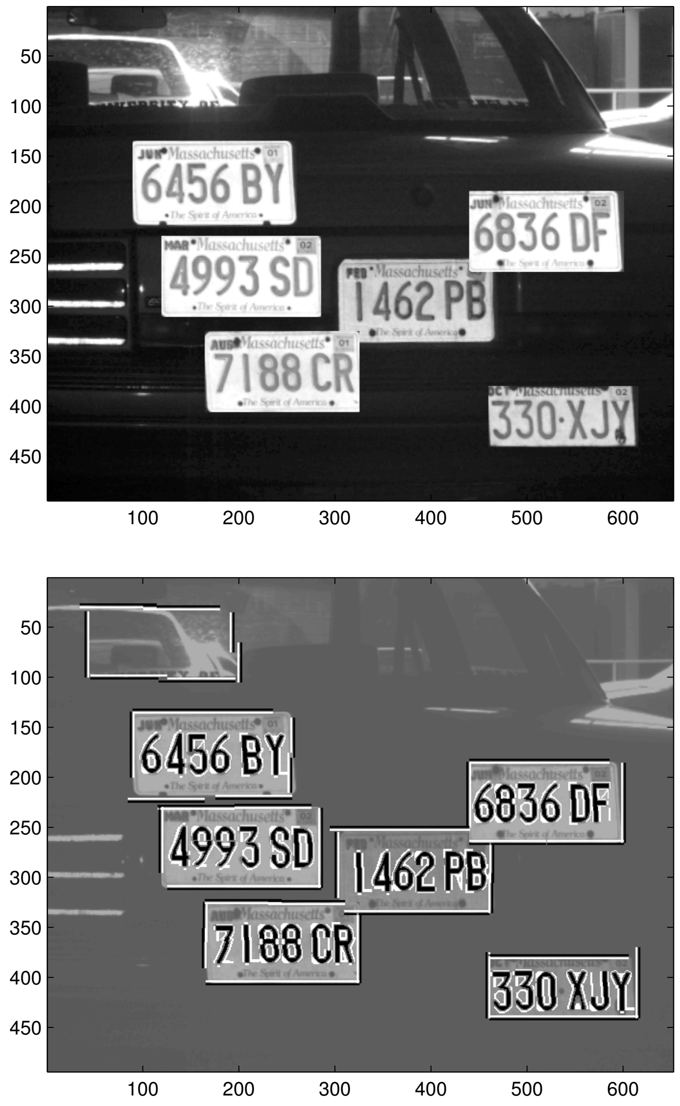


Figure 4.103: Synthesized image and its parse with string and boundary only

Note: Sometimes it is unavoidable to have a false positive boundary in a cluttered background. But is is very rare to have a false positive string right inside a false positive boundary simultaneously, due to the relational binding of both boundary and string hierarchical structures. Multiple license plates with partial overlap in one scene is allowed and no false positive license plate ever occur.

4.4.5.2 Demonstration of Scene Parsing

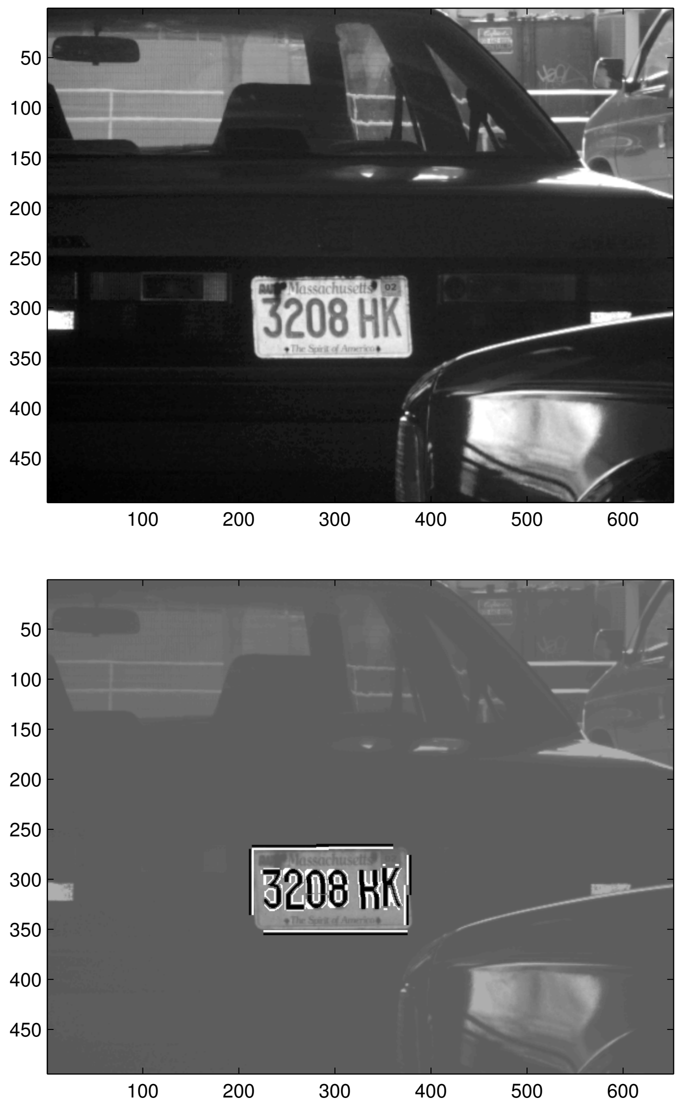
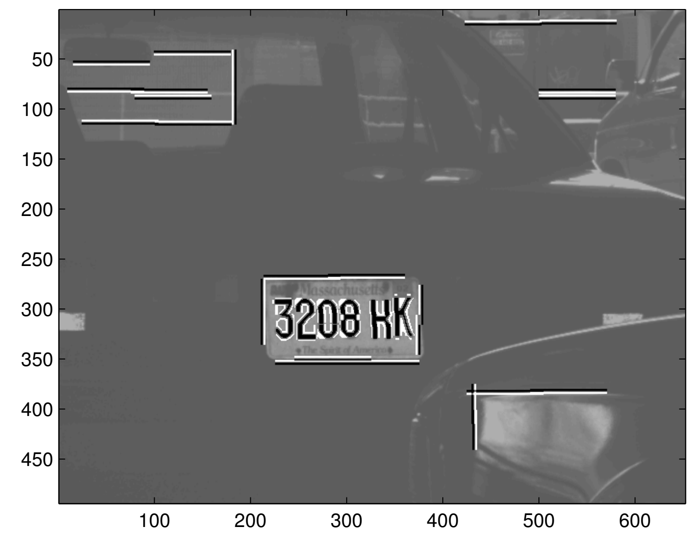


Figure 4.104: original image and its parse with top 1 object



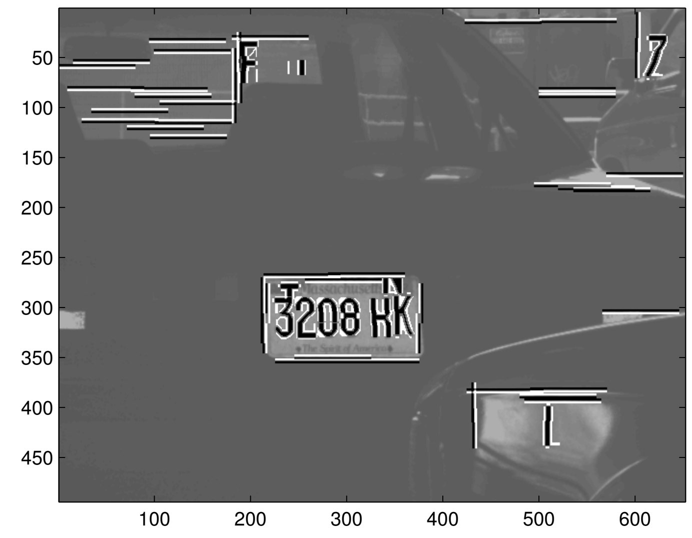
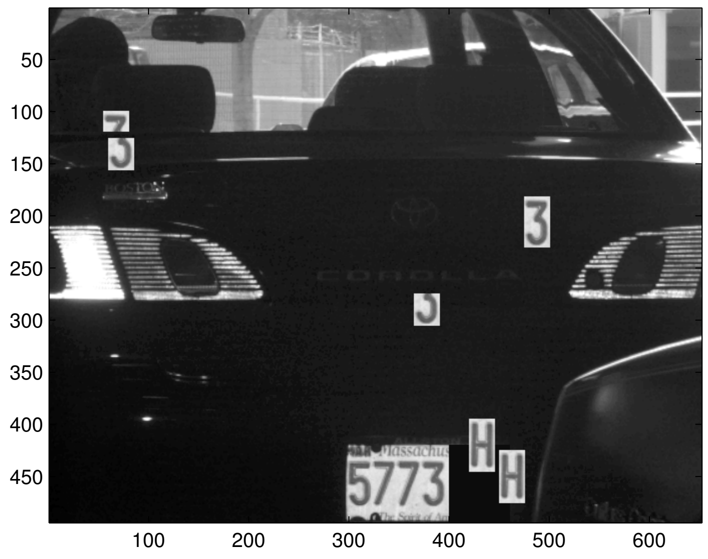


Figure 4.105: its parses with top 10 and top 30 objects

As expected, the top object in the above scene is the license plate, followed by long lines and L-junctions, then followed by false positive characters (composed of often-occuring parts) like “1,” “7,” “F,” “L,” etc. and terminal parts. This parsing process can go on and on until no more objects could increase the total bits gain. Most of the added objects at a later stage are terminal parts.



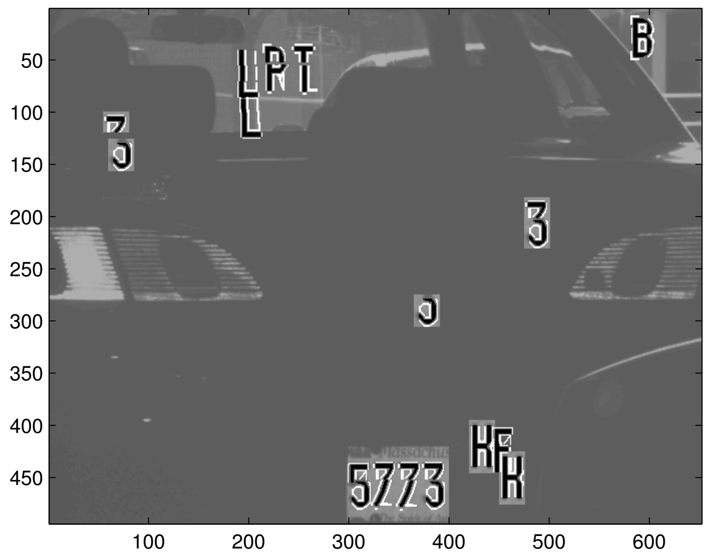


Figure 4.106: Synthesized image and its parse with top 10 string-related objects only. The top five objects are “5773”,“HF”,“LPT”,“3” and “H”.

Note: In Figure 4.106, as desired, no string object is formed due to the poor composition between partial strings. The parsed scene does contain two false positive partial strings "HF" and "LPT." The former one is understandable; it is caused by the artificial gap coming from the two "H" (introduced when we synthesize the image) and the template form of character "F." It is a little bit surprising as to why we had "LPT." After zooming into the region where "LPT" covers, we found "L" and "T" are both properly formed with bits gain above 50, which is conceivable; the region where "P" stands looks quite flat. It seems this "LPT" would be better interpreted as "L" and "T" separately. After further zooming into the "P" region and some preprocess to blow up its contrastness (i.e., thresholding this flat-looking patch with its median), we can see that it is not a constant patch at all in the rank sense (see Figure 4.107). It looks like a poor "P," which is consistent with the computational result. In fact, the bits gain of "P" on such data is -5, which is much bigger than the bits gain on constant patch (about -30). Since "LPT" is nicely composed, its relational bits gain for composition at partial string layer is above 10, which exceeds the data bits loss of "P." It is therefore a justifiable false positive, even though we are not happy that it exists. The main reason for this unpredictable "LPT" is the robustness of the rank sum scheme, which only cares about the relative rank of gray levels of certain pixels. This suggests that for detection purposes, the data model is not enough on its own; it has to be complemented with a strong prior (i.e., Markov backbone and the non-Markovian ratio) to remove the false positive plates (but not necessary false positive partial strings or simpler objects).

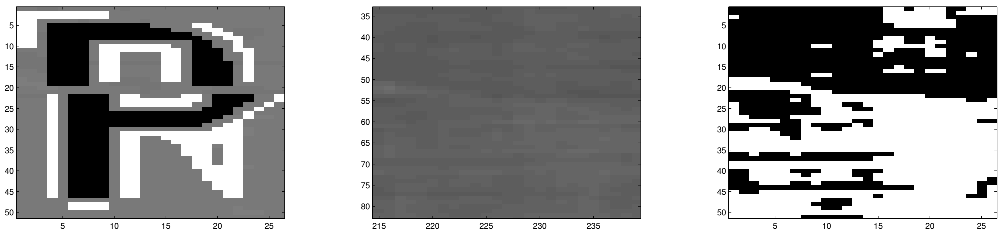


Figure 4.107: Detected false positive "P", its corresponding region, its binary contrast

After we make the constant-looking region between "L" and "T" constant (i.e., its median), we parsed the scene again. Not surprising, there is no character in that region again. Character "1" occurs due to the artificial boundary.

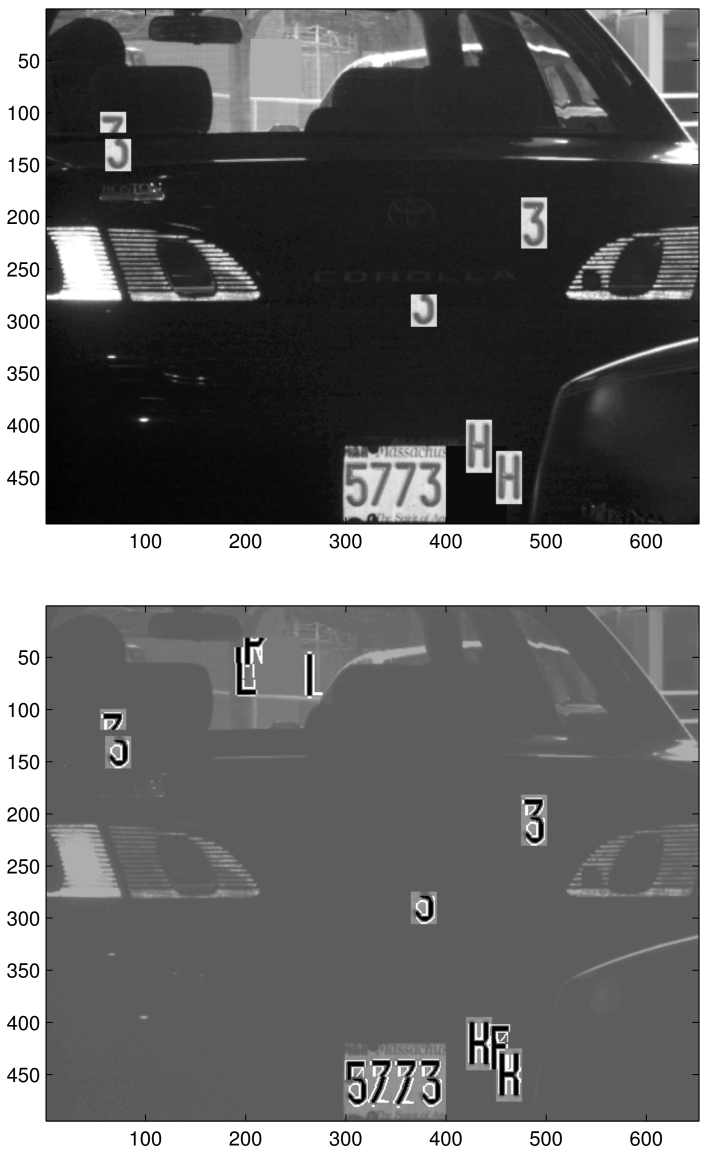


Figure 4.108: Synthesized image (with constant patch) and its parse with top 10 string-related objects only. The top five objects are "5773," "HF," "3," "H" and "1."

4.4.5.3 Demonstration of all Mistakes

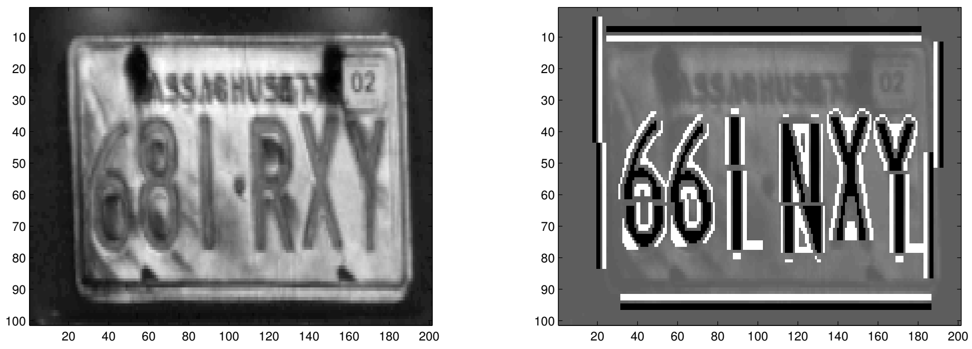


Figure 4.109: Zoomed license region and the wrong output reading

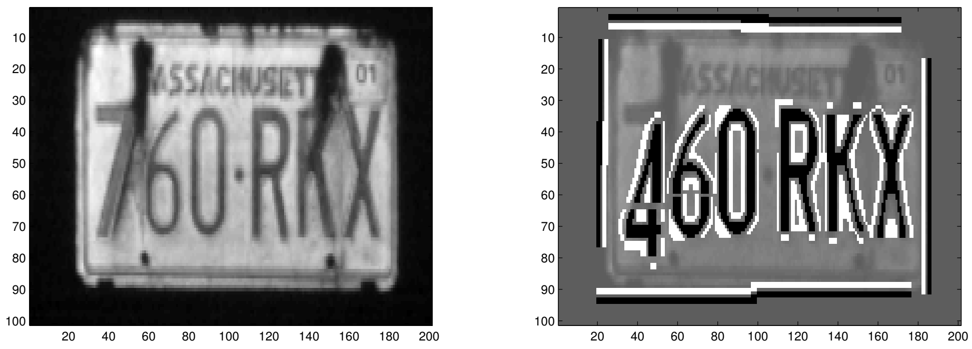


Figure 4.110: Zoomed license region and the wrong output reading

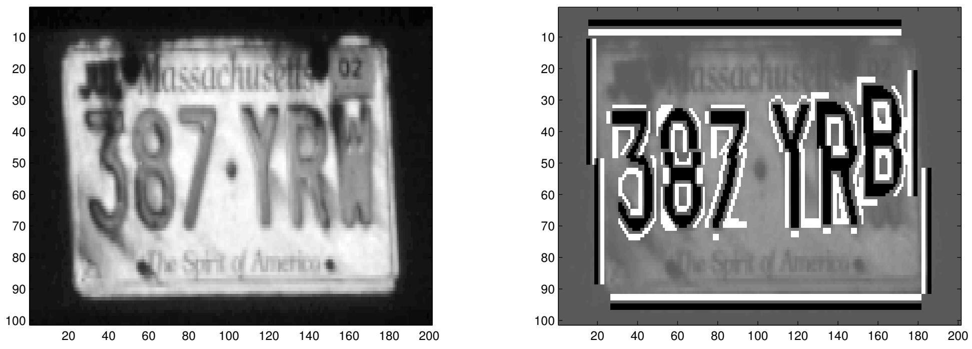


Figure 4.111: Zoomed license region and the wrong output reading

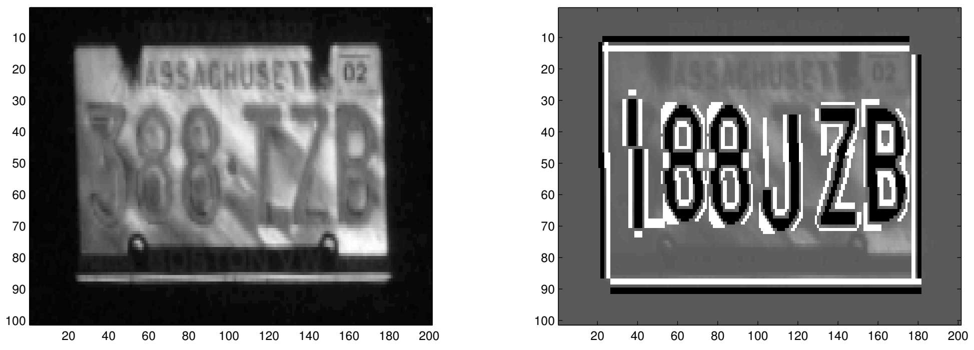


Figure 4.112: Zoomed license region and the wrong output reading

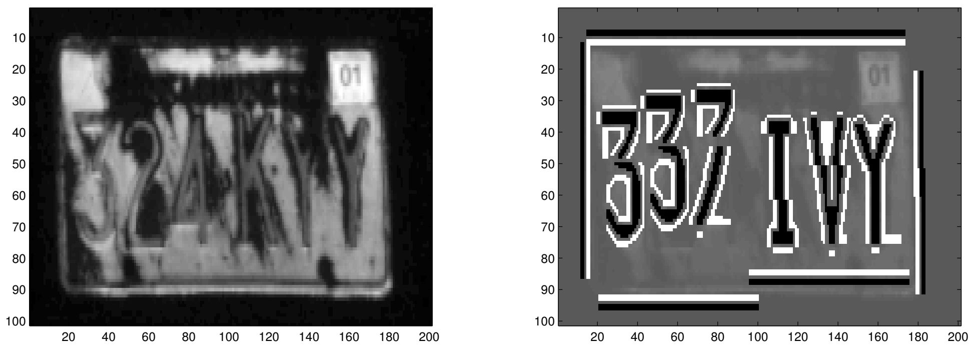


Figure 4.113: Zoomed license region and the wrong output reading

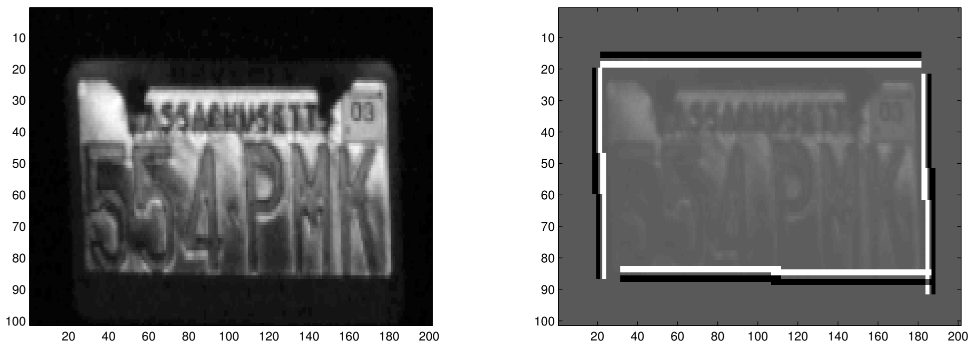


Figure 4.114: Zoomed license region and the wrong output reading

Note that the readings' overall positions are correct, thanks to the relational binding. Combination of the weakness of rank sum scheme, improper parameter setting of the prior model and limited computational power is the main reason for those failures. For deatiled analysis, see next section.

4.5 Model Testing and Suggestions (including parameter learning) from Mistakes

Given the mistakes that were made, we want to know whether they are caused by the model or the computational-related issue. To test this, we manually place the right object at the right place, compute its bits gain and compare it with the wrong object we got. If our model is correct, the right object should yield a bigger bits gain; the reason we failed is due to limited computational power. Otherwise, there must be something we can do to improve our model.

Given the fixed hierarchical structure and the template form of the part filters (how to learn them is not covered here), free parameters left in our model are a state probability vector for each brick and parameters in p^c under composition. How to estimate the state probability has already been covered, i.e., applying EM for terminal parts and applying empirical frequency to high layer objects (translation invariance is implicitly used). So the test of our model is mainly a test of the properness of parameters in p^c. It also tests the goodness of the rank sum scheme.

Given our current model, some of the mistakes made are due to the instability of the rank sum scheme (cases where the wrong character occupies the same position as the right one); the rest are due to improper parameters in p^c (cases where relational binding is not strict enough such that the local data bits gain exceeds global relational bits loss) and/or computational reasons (cases where the true one is missed due to the discretization error by only picking the best terminal part out of 5×5 patch, strict testing threshold, small cluster size and number, etc.).

The top two wrong readings in Figures 4.109-4.110 justify the above reasoning quite well and a few suggestions for improvement are given. The following three wrong readings in Figures 4.111-4.113 share the similar reasoning and the last one in Figure 4.114 is just too noisy to pass the tests for parts.

Consider license reading 681 RXY in Figure 4.109 (Output reading: 661 NXY).
Reason of mistake: Due to the dirt in "8" and "R," the top character candidate at the "8" position is "6," at the "R" position is "N," based on the rank sum scheme.

As expected, the true "8" and "R" are candidates in the list and they are only 2 and 4 bits less than the top ones respectively. Since the wrong characters are right on top of the true ones with decent relational binding, there is not much we can do other than improving the rank sum scheme.

Rank sum statistic sometimes behaves unpredictably under noisy setting. We could try to improve this by incoporating gray level data into the model or applying dense template matching for the confusing candidates. More computational power is undoubtably needed to make this feasible.

Consider license reading 760 RKX in Figure 4.110 (Output reading: 460 RKX).
Reason of mistake: Due to the shadow along "7," rank sum gives us two candidate objects: right-shifted "7" and downshifted "4." Since the right-shifted "7" is too close to the neighboring "6" (in fact, it overlaps with "6"), the relational bits gain of the 3-digit partial string falls below the threshold. But the "460" does pass the threshold.

Looking close into the upper and lower parts of the right-shifted "7," we found their active states are both 21, i.e., the upper-right corner of the corresponding 5×5 patch. Since we only pick the best terminal part from each patch, there is no chance we could have a "7" at the correct position. If we manually put a "7" at the correct position, we have bits gain about 3, compared to 28 bits gain of shifted "7" and 25 bits gain of "4." Looking into the relational bits gain of the 3-digit partial string, the true "760" is only a few more bits better than "460."

The above observations suggest that if we are given more computational power (such that the correct “7” is picked as a candidate) and strict relational binding for partial string and string, then “760 RKX” at the correct position may give us bigger overall bits gain than “460 RKX.” A second suggestion is that given the fact that the horizontal distance between two neighboring characters in a partial string is much more variable than their vertical distance, we could assign a smaller (bigger) σ in p^c characterising the vertical (horizontal) distance and set two thresholds accordingly. Without any change in computational power, we can get the reading right, except the chosen “7” is right-shifted a little. A third suggestion is to introduce more interest points (like lower corners) and relational bindings (like lower corners of characters in a reading should also be aligned), since we can see the lower corners of “4” is poorly aligned with others.

Recall the scene-parsing demonstration (see Figure 4.106). We always get some false positive characters like “L,” “1,” “T,” etc. and false positive partial strings composed of them. This is mainly due to their often-ocurring parts. For these objects, even stricter binding rules should be applied.

It is generally true that we should introduce more relational bindings, pose not-too-loose soft/hard constraints (i.e., standard deviation/threshold), and make full use of the knowledge we have about these relationships.

In a word, lack of computational power combined with improperness of the model (parameters and rank sum scheme) contributes to those mistakes. Most of the time, the two factors are mixed together; it’s hard to correct the mistakes by adjusting only one of them. But with or without the extra computational power, strengthening the performance of rank sum, introducing dense and better relational bindings, and careful estimating relevant parameters have their own meanings.

4.6 Appendix

4.6.1 Terminal Parts for the Characters

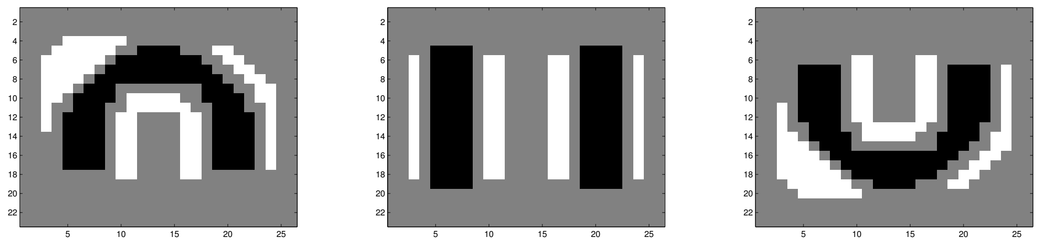


Figure 4.115: Part 1, Part 2, Part 3

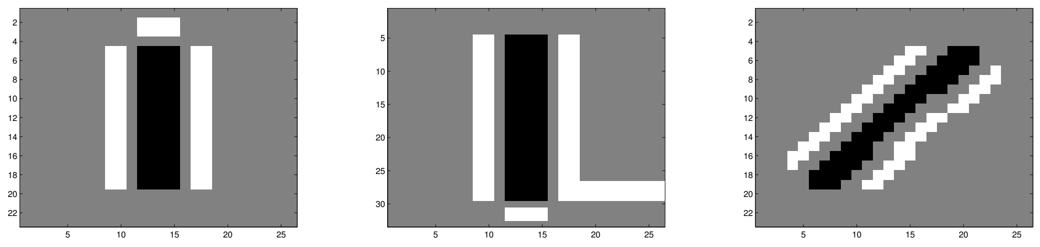


Figure 4.116: Part 4, Part 5, Part 6

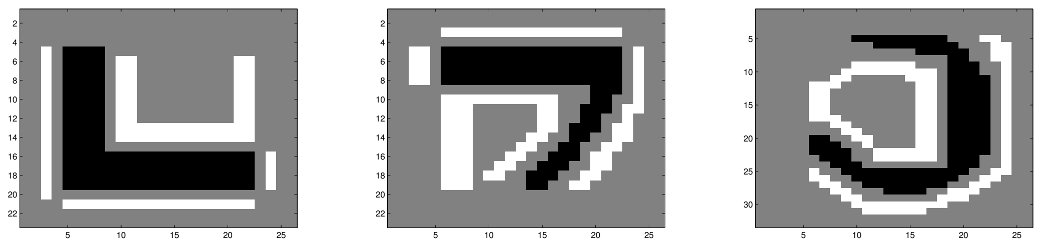


Figure 4.117: Part 7, Part 8, Part 9

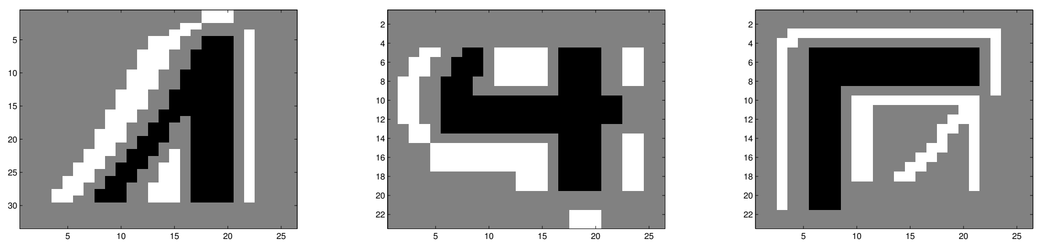


Figure 4.118: Part 10, Part 11, Part 12

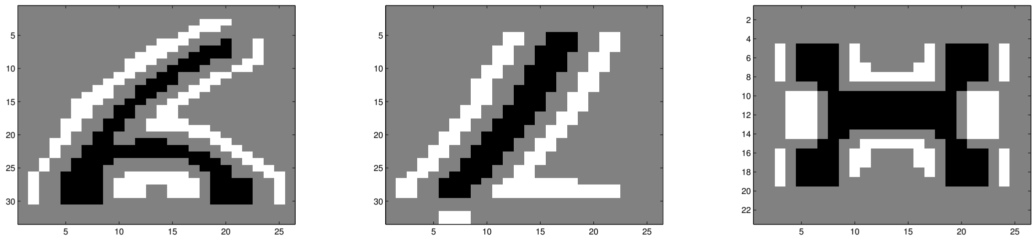


Figure 4.119: Part 13, Part 14, Part 15

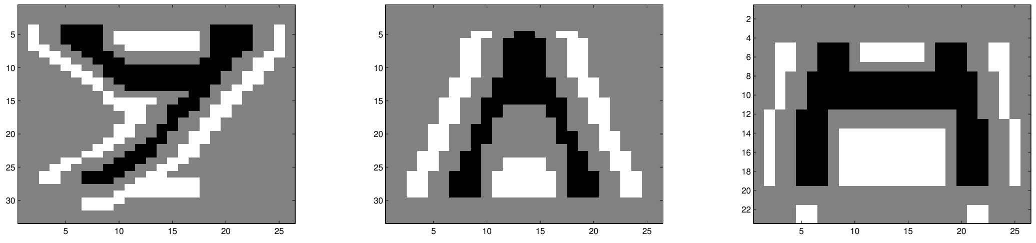


Figure 4.120: Part 16, Part 17, Part 18

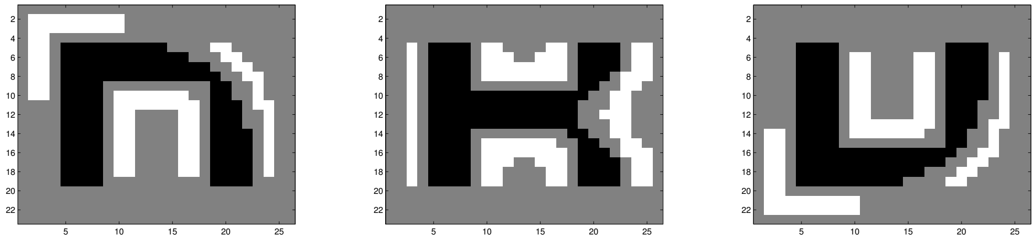


Figure 4.121: Part 19, Part 20, Part 21

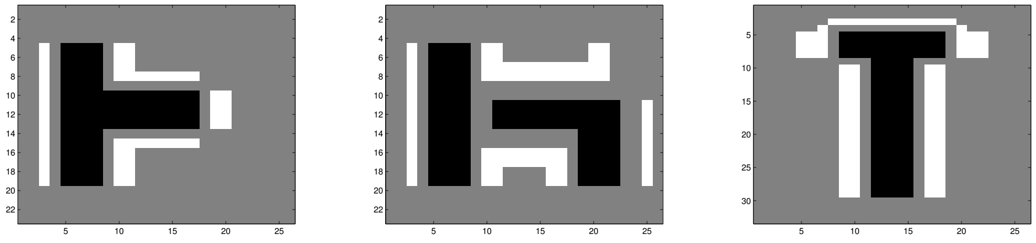


Figure 4.122: Part 22, Part 23, Part 24

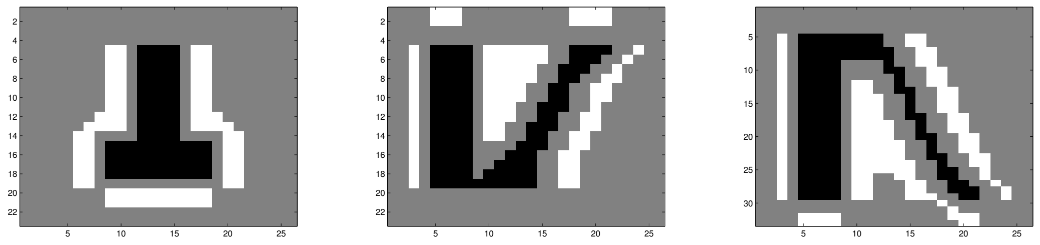


Figure 4.123: Part 25, Part 26, Part 27

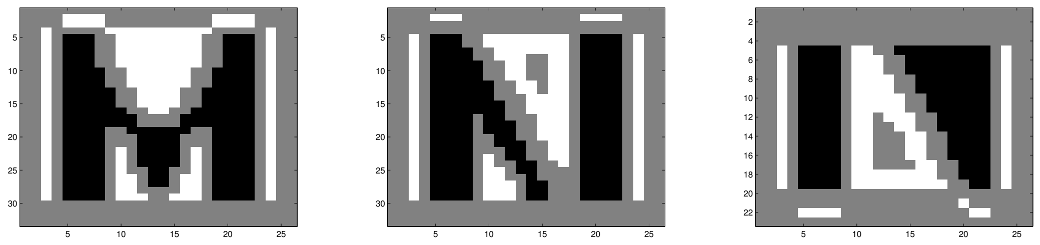


Figure 4.124: Part 28, Part 29, Part 30

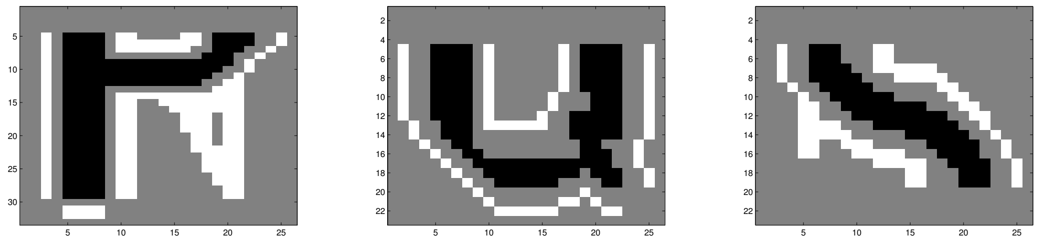


Figure 4.125: Part 31, Part 32, Part 33

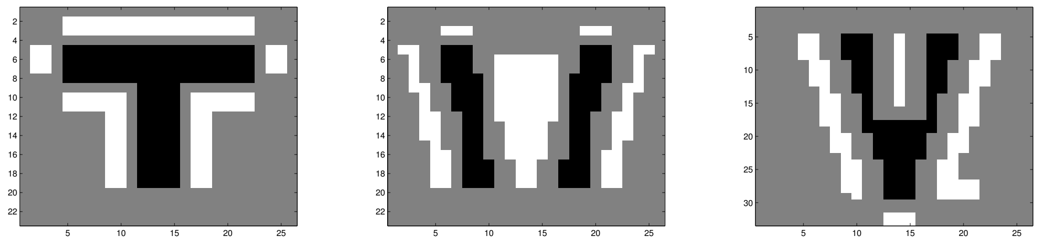


Figure 4.126: Part 34, Part 35, Part 36

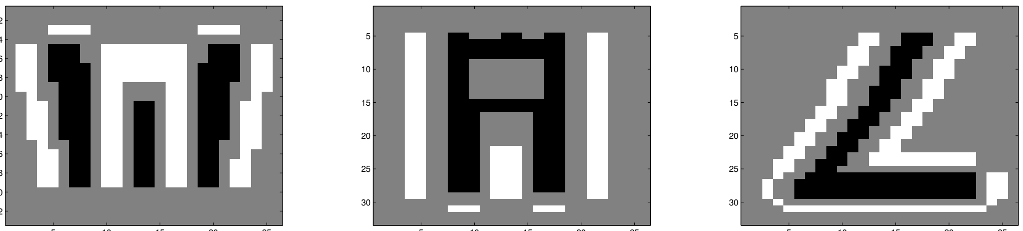


Figure 4.127: Part 37, Part 38, Part 39

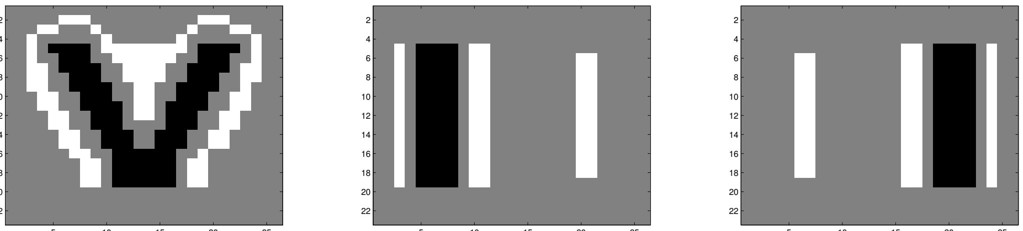


Figure 4.128: Part 40, Part 41, Part 42

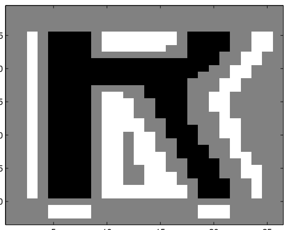


Figure 4.129: Part 43

4.6.2 Terminal Parts for the Boundary

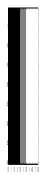


Figure 4.130: short vertical line 1

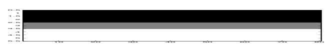


Figure 4.131: short horizontal line 1

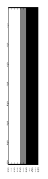


Figure 4.132: short vertical line 2

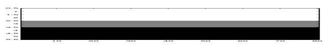


Figure 4.133: short horizontal line 2

4.6.3 the Context-free Rules

String-related:

Digit-0→Part 1 + Part 2 + Part 3

Digit-1→Part 4 + Part 5

Digit-2→Part 1 + Part 6 + Part 7

Digit-3→Part 8 + Part 9

Digit-4→Part 10 + Part 11

Digit-5→Part 12 + Part 9

Digit-6→Part 13 + Part 3

Digit-7→Part 8 + Part 14

Digit-8→Part 1 + Part 15 + Part 3

Digit-9→Part 1 + Part 16

Letter-A→Part 17 + Part 18

Letter-B→Part 19 + Part 20 + Part 21

Letter-C→Part 1 + Part 41 + Part 3

Letter-D→Part 19 + Part 2 + Part 21

Letter-E→Part 12 + Part 22 + Part 7

Letter-F→Part 12 + Part 22 + Part 41

Letter-G→Part 1 + Part 23 + Part 3

Letter-H→Part 2 + Part 20 + Part 2

Letter-I→Part 24 + Part 25

Letter-J→Part 42 + Part 42 + Part 3

Letter-K→Part 26 + Part 27

Letter-L→Part 41 + Part 41 + Part 7

Letter-M→Part 28 + Part 2

Letter-N→Part 29 + Part 30

Letter-O→Part 1 + Part 2 + Part 3

Letter-P→Part 19 + Part 31

Letter-Q→Part 1 + Part 2 + Part 32

Letter-R→Part 19 + Part 43

Letter-S→Part 1 + Part 33 + Part 3

Letter-T→Part 34 + Part 5

Letter-U→Part 2 + Part 2 + Part 3

Letter-V→Part 35 + Part 36

Letter-W→Part 37 + Part 38

Letter-X→Part 40 + Part 17

Letter-Y→Part 40 + Part 5

Letter-Z→Part 8 + Part 39

General digit→Digit-0 or Digit-1 or ... Digit-9

General letter→Letter-A or Letter-B or ... Letter-Z

2-Letter Partial-string→ General letter + General letter

3-Digit Partial-string→ General digit + General digit + General digit

3-Letter Partial-string→ General letter + General letter + General letter

4-Digit Partial-string→ General digit + General digit + General digit + General digit

String→4-Digit + 2-Letter or 3-Digit + 3-Letter

Boundary-related:

Long vertical line 1→Short vertical line 1 + Short vertical line 1

Long vertical line 2→Short vertical line 2 + Short vertical line 2

Long horizontal line 1→Short horizontal line 1 + Short horizontal line 1

Long horizontal line 2→Short horizontal line 2 + Short horizontal line 2

L-shape 1→Long vertical line 1 + Long horizontal line 1

L-shape 2→Long vertical line 2 + Long horizontal line 2

Boundary→L-shape 1 + L-shape 2

License-related:

License plate→String + Boundary

Chapter 5

Conclusion and Future Directions

Conclusion

The approach we propose involves great deal of computation and modeling. Although we covered some aspects of parameter learning, the main architecture of the composition system, including the selection of bricks, children sets and attribute functions, is not systematically learned. It is tempting to think the structure is learned in a bottom-up fashion, layer by layer, maybe based on some principle like "suspicious coincidence" suggested by Barlow [3]. To really understand the learning process, further evidence from cognitive science and neuron science community is needed, since we believe such learning bears a great deal of similarity with early stage learning in children. For now, we just represent the knowledge in a hardwired fashion, assuming it is already learned by an adult.

We believe "hierarchy of reusable parts" points us in a direction to bridge the ROC gap which persistently occurs in comparison of human and machine performance in vision. As we know, with quite a bit of training, current machines can achieve reasonable detection rate at an allowable false positive rate, but every percentage of improvement toward perfect detection (i.e. no missing target) is overwhelmed by massive false positive targets. The underlying reason is that there was never a proper background model. In some sense, people all try to separate background from foreground; they model the general background as simple as white noise or as complicated as MRF depending on the assumption, but model the foreground object in a totally different manner, one not compatible with the background model. We all know the mistakes (false positives) occur at the most ambiguous region (the cluttered region) and image patch in this region looks like everything including the target to the machine if you have to set one threshold for a test of object versus non-object. Not hard to notice, those cluttered regions are made of the same parts as the foreground. This prevailing sharing phenomena paired with predominant false detections in this region suggests an object equipped with its own background model, that is, a model that accommodates both object and background which consist of the same reusable parts, in a uniform way. The composition system embodies this idea by defining an interpretation as a complete sub graph, with a forest of trees rooted at different levels (lower levels usually refer to background or simple objects and higher levels refer to objects of interest). Under this framework, we have shown theoretically and experimentally that the ROC gap can be narrowed greater than ever.

Future Directions

1. Build a composition machine for deformable object recognition.

In the (fixed-font) license-reading application, we start the terminal layer with parts that are not quite primitive (see appendix), since they can still be naturally decomposed into bars with different orientations, curvelets, etc. It is acceptable that we don't add extra layers in this setting due to the fact that there are enough contextual bindings (i.e. quite a few layers in the compositional system) for this fixed-font task. But for a general task involving deformable objects, say handwriting recognition, current character parts are too complicated to be reused for this task. In this new setting, simpler parts (like linelets and curvelets) have to be employed to encourage reusability, more robust attribute functions (like relative scale and relative angle) have to be considered to accommodate the variability, and more attention is paid to design the (composed and null) distributions over the attributes. Besides, more layers are usually employed to represent deformable objects in the compositional system, full-grown version of current depth-first search algorithm needs to be implemented. The bottom line is: a certain amount of extra modeling has to be done for a generalized application, but the overall architecture of the composition system doesn't change.

2. Automate the generation of application-dependent compositional system.

Learning the architecture of a composition system is challenging at the current stage, but building an application-dependent compositional machine in a semi-automatic fashion is doable. Consider a software package equipped with a user-friendly interface, through which a sequence of general questions are asked to a specific user with his own application in mind. The typical asked questions are: What is your object? What is it composed of? How is it composed of, horizontally, vertically, or in both directions? What is its RF? What is its attribute? Do you want to further decompose the current object? They are asked in a particular order, usually in a top-down manner. For the parts on the terminal layer, the software contains a library of reusable classes, each capturing a reusable part and its related operations. The user can pick the existed terminal parts from the GUI or define his own parts and add them into the system. Provided with the user's chosen parts and answers to the above questions, the software can quickly build the compositional machine. Then by training over a bunch of raw images to get a better estimation of a few involved parameters, the compositional machine is well-tuned for the application. Finally, the user can press different buttons on the interface to invoke different functions, such as object detection and scene parsing. You can stop the function at any time, and the typical output will be the test image with each of its pixels labeled.

The software can even be generalized to deal with video information, as long as a proper composition machine is built following properly asked questions (Note: in this case, object brick extends to event brick, attribute set includes causal relation and the same architecture holds) and a fast search algorithm is implemented. Then, there will be one more functionality (object tracking) built into the system.

Bibliography

[1] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multi-class shape detection. *IEEE Trans. PAMI*, 2004.

[2] Y. Amit and A. Trouve. Pop: Patchwork of parts models for object recognition. Technical report, University of Chicago, 2004.

[3] H. Barlow. What is the computational goal of the neocortex? In Christof Koch and Joel L. Davis, editors, *Large-Scale Neuronal Theories of the Brain*, pages 1–22. MIT Press, Cambridge, 1994.

[4] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[5] E. Bienenstock, S. Geman, and D. Potter. Compositionality, mdl priors, and object recognition. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 838–844. MIT Press, 1997.

[6] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK., 1995.

[7] G. Blanchard and D. Geman. Hierarchical testing designs for pattern recognition. *Annals of Statistics*, 33:1155–1202, 2005.

[8] E. Borenstein and S. Ullman. Class specific top down-segmentation. In *Proc. ECCV*, pages 110–122, 2001.

[9] L. Breiman. Arcing classifiers. *Annals of Statistics*, 26(3):801–849, 1998.

[10] N. Chomsky. *Aspects of the Theory of Syntax.* MIT Press, 1965.

[11] David Crandall, Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *Proc. CVPR*, pages 10–17, 2005.

[12] A. Brandt E. Sharon and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. *IEEE Conf. on Computer Vision and Pattern Recognition*, 1:469–476, 2001.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[14] K. S. Fu. *Syntactic Methods in Pattern Recognition.* Academic Press, New York, 1974.

[15] K Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1:119–130, 1988.

[16] S Geman. On the formulation of a composition machine. Technical report, Division of Applied Mathematics, Brown University, 2005.

[17] S. Geman and M. Johnson. Probability and statistics in computational linguistics, a brief review. In Mark Johnson, Sanjeev Khudanpur, Mari Ostendorf, and Roni Rosenfeld, editors, *Mathematical foundations of speech and language processing*, volume 138, pages 1–26. Springer-Verlag, 2003.

[18] S. Geman, K. Manbeck, and E. McClure. Coarse-to-fine search and rank-sum statistics in object recognition. Technical report, Division of Applied Mathematics, Brown University, 1995.

[19] S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, LX:707–736, 2002.

[20] U. Grenander. *General Pattern Theory: A Study of Regular Structures.* Oxford University Press, 1993.

[21] M. Harrison. *Discovering Compositional Structure.* PhD thesis, Division of Applied Mathematics, Brown University, 2005.

[22] S. H. Huang. *Compositional approach to recognition using multi-scale computations.* PhD thesis, Division of Applied Mathematics, Brown University, 2001.

[23] W. T. Freeman J. Yedidia and Y. Weiss. Understanding belief propagation and its generalizations. *International Joint Conference on Artificial Intelligence*, 2001.

[24] S. Krempp, D. Geman, and Y. Amit. Sequential learning of reusable parts for object detection. Technical report, Johns Hopkins University, 2003.

[25] P.S. Laplace. *Esssai philosophique sur les probabilités.* New York, 1965. Translation of Truscott and Emory.

[26] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157. IEEE Press, 1999.

[27] M. Welling M. Weber and P. Perona. Unsupervised learning of models for recognition. *In Proc. of ECCV*, pages pp. 18–32, 2000.

[28] M. Riesenhuber and T. Poggio. Models of object recognition. *Nature Neuroscience*, 3 supp.:1199–1204, 2000.

[29] J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14(3):1080–1100, 1986.

[30] S. Roth and M.J. Black. Fields of experts: A framework for learning image priors. *IEEE Conf. on Computer Vision and Pattern Recognition*, II:860–867, 2005.

[31] E. Sali S. Ullman and M. Vidal-Naquet. A fragment-based approach to object representation and classification. *IWVF*, pages pp. 85–102, 2001.

[32] M. Stone. Cross-validatory choice and assessment of statistical predictors (with discussion). *J.R. Statist. Soc.*, B36:111–147, 1974.

[33] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. CVPR*, pages 762–769, 2004.

[34] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, New York, 1982.

[35] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, Berlin, 1995.