Linda consists of a few simple operations that embody the tuple space model of parallel programming. Adding these tuple-space operations to a base language yields a parallel programming dialect. Use of Linda is growing

The fact that senders in Linda needn't know anything about receivers and *vice versa* is central to the language. It promotes what we call an uncoupled programming style. When a Linda process generates a new result that other processes will need, it simply dumps the new data into tuple space. A Linda process that needs data looks for it in tuple space. In message passing systems, on the other hand, a process can't disseminate new results without knowing precisely *where* to send them. While designing the data generating pro-

The merging problem in Prolog-derived concurrent languages is well-known. Some researchers argue that the solution is simply to add a new *merge* primitive. This solution points to a deeper problem, however. The

The Linda operating system environment we're now building accommodates multiple first class tuple spaces [20]. A tuple space is created with certain attributes—for example, some tuple spaces are persistent, and persistent tuple spaces constitute the file system. Whole tuple spaces can be treated as single objects: they can be suspended, archived, reactivated or snapshotted *en masse*. As always, a tuple space may contain active as

In this paper we present **DEDALUS**, a foundation language for programming and reasoning about distributed systems. DEDALUS reduces to a subset of Datalog [30] with negation, aggregate functions, successor and choice, and admits an explicit representation of time into the logic language. We show that DEDALUS provides a declarative foundation for the two signature features of distributed systems: mutable state, and asynchronous processing and communication.

DEDALUS$_0$ programs are intended to capture temporal semantics. For example, a fact, $\mathtt{p}(C_1 \ldots C_n, C_{n+1})$, with some constant C_{n+1} in its time suffix can be thought of as a fact that is true "at time C_{n+1}". Deductive rules can be seen as *instantaneous* statements: their deductions hold for predicates agreeing in the time suffix and describe what is true "for an instant" given what is known at that instant. Inductive rules are *temporal*—their consequents are defined to be true "at a different time" than their antecedents.

Time is a device that was invented to keep everything from happening at once.[2]

for each timestep t, the temporal evaluation yields a database that corresponds to the minimal model of the original DEDALUS$_0$ program with the `successor` relation truncated to the prefix ending at t.

For a query q, the system is searching for documents d which imply the query logically, i.e. for which the logical formula $q \leftarrow d$ is true. Due to the intrinsic vagueness and imprecision of IR, a logic that allows for uncertainty reasoning should be used. In [Rijsbergen 86],

Probabilistic Datalog is an extension of stratified Datalog (see e.g. [Ullman 88], [Ceri et al. 90]). On the syntactical level, the only difference is that with ground facts, also a probabilistic weight may be given, e.g.

```
0.7 indterm(d1,ir).  0.8 indterm(d1,db).
```

we can transform P_E into the corresponding set P_E of (deterministic) Datalog clauses, for which there exists a least Herbrand model $HL(\bar{P}_E)$. The interpretation of P_E is a set of possible worlds, where each world is a subset of $HL(P_E)$.

— The rule-based approach allows for easy formulation of retrieval models for specific or novel applications, like e.g. combination with a thesaurus or retrieval in hypertext bases or hierarchically structured documents.

To solve these problems, we propose a new interaction technique for drawing, *interactive beautification*. Interactive beautification is a technique for rapid construction of geometric diagrams (an example is shown in Figure 1) without using any editing commands or special interaction modes. Interactive beautification can be seen as an extension of free stroke vectorization [7] and diagram beautification [18]. It receives a user's free stroke and beautifies the stroke considering various geometric constraints among segments. The intuitiveness of the technique allows novice users to draw such precise diagrams rapidly without any training.

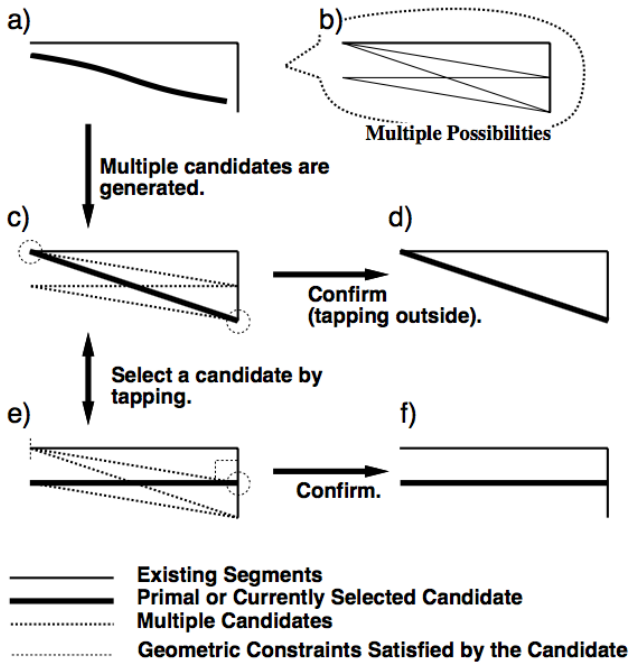


Figure 7: Interaction with multiple candidates: the user can select a candidate by tapping on it, and satisfied constraints are visually indicated.

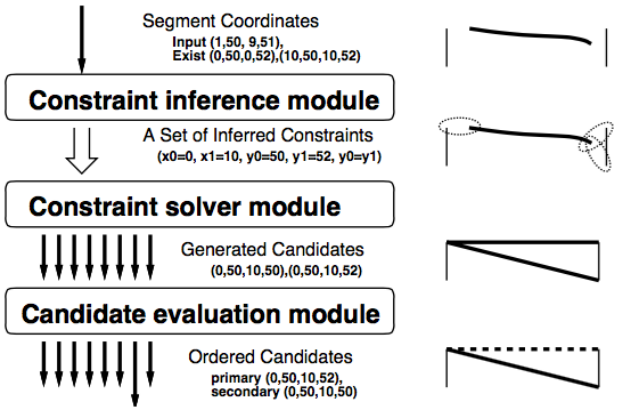


Figure 10: Sturcture of the beautification routine

field. We introduce the notion of *mass*, *distance*, *gravitational force*, *repulsive force*, and *inertia* of objects to define a metric space for the field, so that objects migrate to their (sub)optimal (or satisfactory) locations. We also propose a communication model called *assimilation/dissimilation model*, where communication is represented as the assimilation/dissimilation of messenger objects and their migration.

In this paper, we propose a new computing model called *computational field model*, or *CFM* for short. CFM is a computing model for solving many large and complex problems of different purposes in an open-ended distributed environment. We understand that solving a problem implies mutual effects between the computational field and the problem. That is, problems themselves change or affect the environment itself. Or, we can even say that problems are a part of the computing environment. In this sense, we share the same view as described in [Huberman 88].

We then introduce the notions of *distance* between objects and the *mass* of an object to the model in order to form a metric space. We define *gravitational force*, *repulsive force*, and *inertia*, metaphorically to dynamics. Using these measures, objects migrate to their (sub)optimal (or satisfactory) locations during computation (i.e. over their lifetime). We call this *Mass and Distance-based computing* or *MD-based computing* for short.

the objects. Object-oriented computing can be understood as the departure from the microscopic view of computing where computation proceeds by executing an algorithm of a procedure to the macroscopic view where computation proceeds as mutual effects among objects.

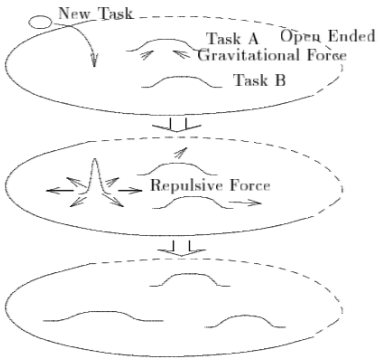


Figure 1: MD-based computing in a computational field

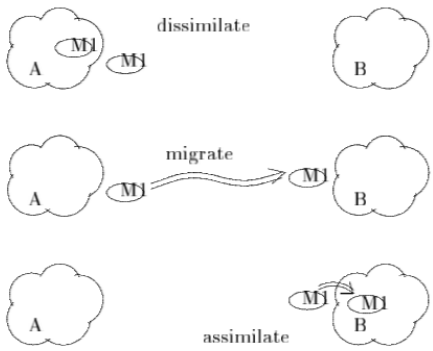


Figure 2: Unidirectional asynchronous communication

Croquet was built to answer a simple question. If we were to create a new operating system and user interface knowing what we know today, how far could we go? What kinds of decisions would we make that we might have been unable to even consider 20 or 30 years ago, when the current operating systems were first created?

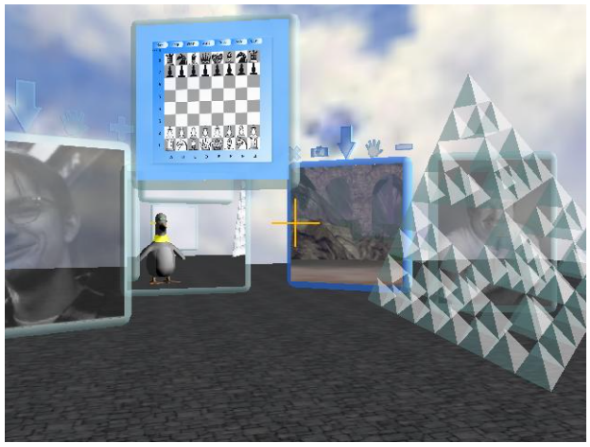


Figure 1. Croquet multi-user environment.

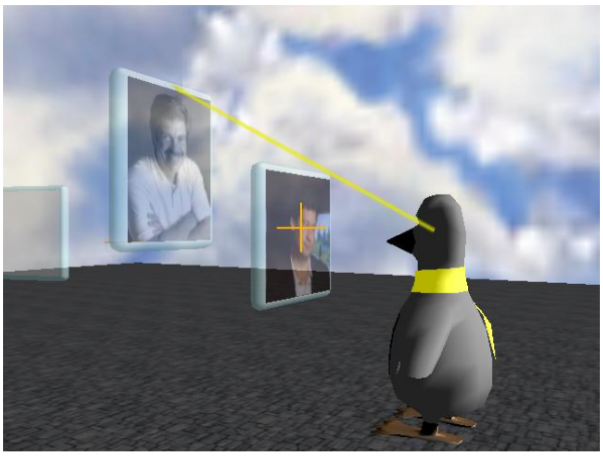


Figure 2: The other user is dragging a window up into the air.

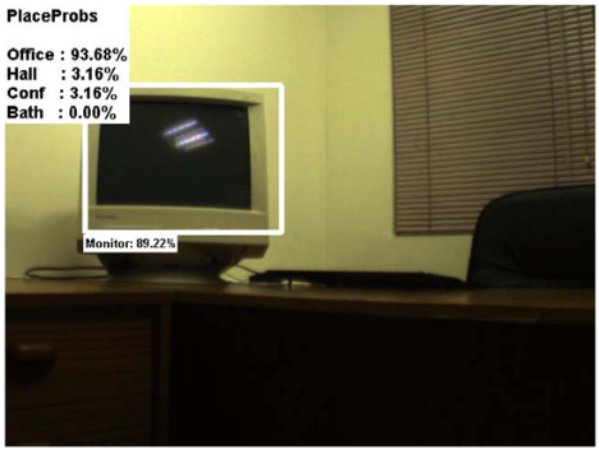
- There are no boundaries in the system. We are creating an environment where anything can be created; everything can be modified, all while still inside the 3D world. There is no separate development environment, no user environment. It is all the same thing. We can even change and author the worlds in collaboration with others *inside them while they are operating* .

- A coordinated universal timebase embedded in communications protocol.
- Replicated, versioned objects – unifying replicated computation and distribution of results.
- Replication strategies – that separate the mechanisms of replication from the behavioral semantics of objects.
- Deadline-based scheduling extended with failure and nesting.
- A coordinated "distributed two-phase commit" that is used to control the progression of computations at multiple sites, to provide resilience, deterministic results, and adaptation to available resources. Uses distributed sets.
- Time-synchronized I/O. Input and output to real-time devices are tied to coordinated universal time.

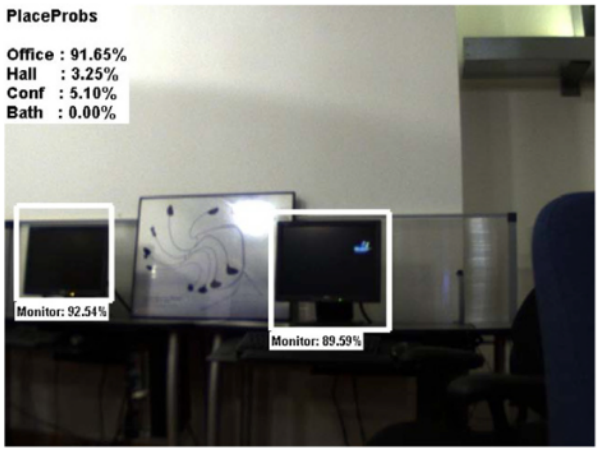
Following the motivations above, in this paper we propose a new technique for visual indoor scene recognition using a mobile robot. As distinguishing features, our approach is based on three main features. First, a probabilistic hierarchical representation that uses common indoor objects, such as Doors or furniture, as an intermediate semantic representation. Using this representation, we associate low-level visual features to objects by training object classifiers, and we associate objects to scenes by learning contextual relations among them. Second, we exploit the embedded nature of a mobile robot by using 3D information to implement a focus of attention mechanism. Using this mechanism, we can use 3D information to discard unlikely object locations and sizes. Third, we also exploit the embedded nature of a mobile robot and ideas from information theory to implement an adaptive strategy to search for relevant objects. Under this strategy, we use sequences of images captured during robot navigation to build a partial belief about the current scene that allow us to execute only the most informative object classifiers.

Recent approaches have achieved good results in scene classification by using intermediate representations and bag-of-words schemes. Fei-Fei and Perona recognize scenes using an intermediate representation that is provided by an adapted version of the Latent Dirichlet Allocation (LDA) model [14]. Bosch et al. [35] and Sivic et al. [36] achieve scene classification by combining probabilistic Latent Semantic Analysis (pLSA) with local invariant features. Lazebnik et al. modify bag-of-words representations by using a spatial pyramid that divides the image into increasingly fine sub-regions with the idea of capturing spatial relations among different image parts [37]. As we mentioned before, these techniques show a significant drop in performance for the case of indoor scenes [15].

In all our tests we use QVGA images (320 × 240 pixels)



(a) DCC-PUC.



(b) CSAIL-MIT.

Table 1
Confusion matrices for compared methods.

Scene	Off. (%)	Hall (%)	Conf. (%)	Bath. (%)
	OM			
Office	**91**	7	2	0
Hall	7	**89**	4	0
Conference	7	7	**86**	0
Bathroom	0	6	0	**94**

The human labeler has erred here:

(a) A Bed is detected.

(b) A TV Monitor is detected.

Fig. 15. Method operating inside a Kitchen.

A central goal of image-based rendering is to evoke a visceral sense of *presence* based on a collection of photographs of a scene. The last several years have seen significant progress towards this goal

- **Where was I?** Tell me where I was when I took this picture.
- **What am I looking at?** Tell me about objects visible in this image by transferring annotations from similar images.

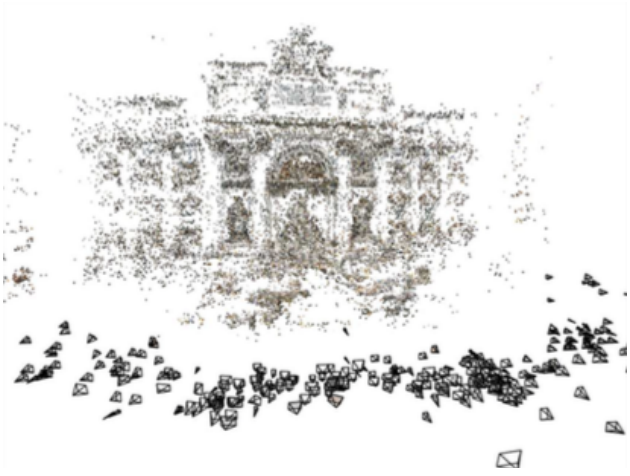
The backbone of our system is a robust structure from motion approach for reconstructing the required 3D information. Our approach first computes feature correspondences between images, using descriptors that are robust with respect to variations in pose, scale, and lighting, and runs an optimization to recover the camera parameters and 3D positions of those features. The resulting correspondences and 3D data enable all of the aforementioned features of our system.

background scene structures. As such, we side-step the more challenging problems of reconstructing full surface models [Debevec et al. 1996; Teller et al. 2003], light fields [Gortler et al. 1996; Levoy and Hanrahan 1996], or pixel-accurate view interpolations [Chen and Williams 1993; McMillan and Bishop 1995; Seitz and Dyer 1996; Zitnick et al. 2004]. The benefit of doing this is that we are able to operate robustly with input imagery that is beyond the scope of previous IBM and IBR techniques.

Sift and RANSAC are documented later

The first step is to find feature points in each image. We use the SIFT keypoint detector [Lowe 2004], because of its invariance to image transformations. A typical image contains several thousand

images, we match keypoint descriptors between the pair, using the approximate nearest neighbors package of Arya, *et al.*[1998], then robustly estimate a fundamental matrix for the pair using RANSAC



- A set of points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$. Each point consists of a 3D location and a color obtained from one of the image locations where that point is observed.
- A set of cameras, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Each camera C_j consists of an image I_j, a rotation matrix $\mathbf{R}_j$, a translation t_j, and a focal length f_j.
- A mapping, Points, between cameras and the points they observe. That is, $\text{Points}(C)$ is the subset of $\mathcal{P}$ containing the points observed by camera c.
- A set of 3D line segments $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ and a mapping, Lines, between cameras and the set of lines they observe.

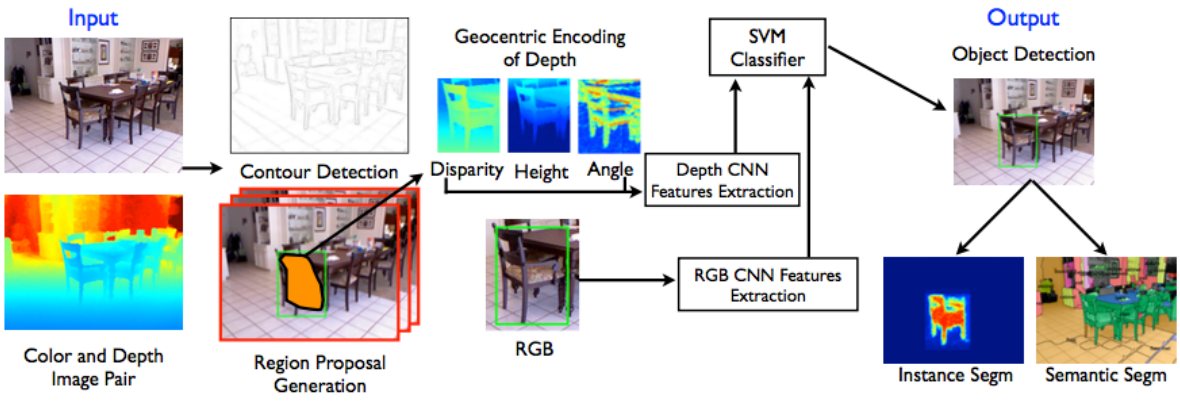


Fig. 1. Overview: from an RGB and depth image pair, our system detects contours, generates 2.5D region proposals, classifies them into object categories, and then infers segmentation masks for instances of "thing"-like objects, as well as labels for pixels belonging to "stuff"-like categories.

In [18], we used *gPb-ucm* [2]

Augmentation with synthetic data: An important observation is the amount of supervised training data that we have in the NYUD2 dataset is about one order of magnitude smaller than what is there for PASCAL VOC dataset (400 images as compared to 2500 images for PASCAL VOC 2007). To address this issue, we generate more data for training and finetuning the network. There

SVM Training: For training the linear SVMs, we compute features either from pooling layer 5 (*pool5*), fully connected layer 6 (*fc6*), or fully connected layer 7

input channels	RGB	RGBD	RGB	RGB	disparity	disparity	HHA	HHA 2x	HHA 15x	HHA 2x	HHA 2x	RGB+HHA 2x	dress--er	garba--ge bin
synthetic data?													1.4	6.6
													16.4	26.7
													18.9	15.7
													30.4	**39.4**
													26.9	32.9
													29.0	37.1
													33.2	38.1
													33.7	**38.3**

The aim of this work is to retrieve those key frames and shots of a video containing a particular object with the ease, speed and accuracy with which Google retrieves text documents (web pages) containing particular words. This paper investigates whether a text retrieval approach can be successfully employed for object recognition.

Each elliptical affine invariant region is represented by a 128-dimensional vector using the SIFT descriptor devel-

3. Building a visual vocabulary

The objective here is to vector quantize the descriptors into clusters which will be the visual 'words' for text retrieval.

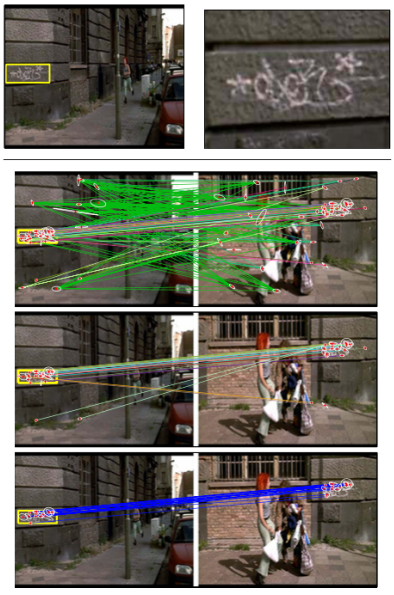


Figure 6: Matching stages. Top row: (left) Query region and (right) its close-up. Second row: Original word matches. Third row: matches after using stop-list, Last row: Final set of matches after filtering on spatial consistency.

The text retrieval analogy also raises interesting questions for future work. In text retrieval systems the textual vocabulary is not static, growing as new documents are added to the collection. Similarly, we do not claim that our vector quantization is universal for all images. So far we have learnt vector quantizations sufficient for two movies, but ways of upgrading the visual vocabulary will need to be found. One could think of learning visual vocabularies for different scene types (e.g. city scape vs a forest).

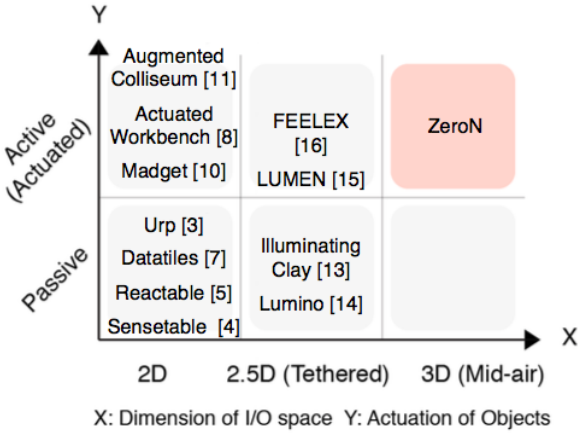


Figure 2. A framework for tabletop tangible interfaces based on the dimension of I/O space and actuation

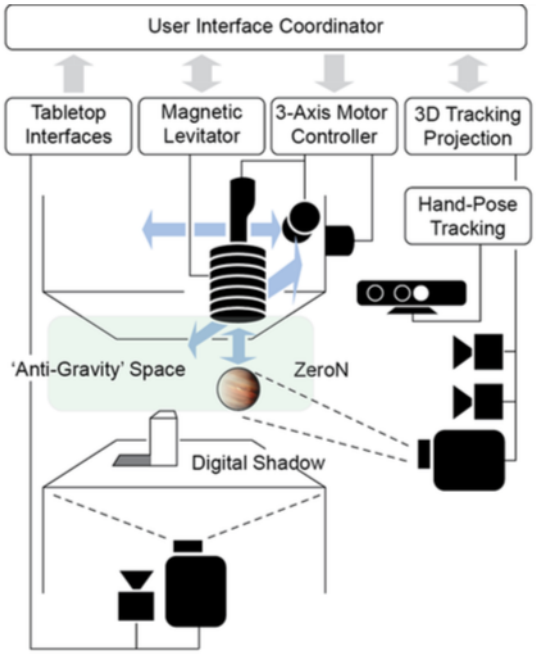


Figure 4. Overview of the ZeroN System

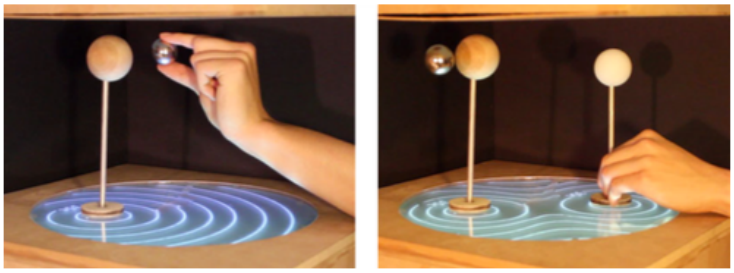


Figure 15. Visualizing 3-body problem.

3D Motion Prototyping

Creating and editing 3D motion for animation is a long and complex process with conventional interfaces, requiring expert knowledge of the software, even for simple prototyping. With record and play-back interaction, users can easily prototype the 3D movement of an object and watch it playing back in the real world. The motion can possibly be mapped to a 3D digital character moving accordingly on the screen with dynamic virtual environments. As a result,

This work presents *GaussBricks* (Figure 1), a system of magnetic building blocks that allow users to construct a tangible form on the displays. Each magnetic building block containing strong magnets is designed simply to facilitate configurable and stable form construction. The physical form con-

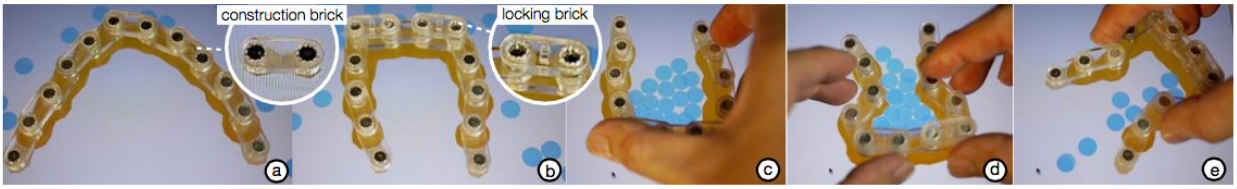


Figure 2. Construct tangible form for physical simulation in the following steps: (a) Use *construction bricks* to construct and shape articulated physical structures on the display. Balls in the simulated gravity bounced away when colliding with the physical structure. (b) Use *locking bricks* to rigidify parts of the construction. (c) Holding the rigid parts to perform spatial operations without affecting the structure. (d) Shape the non-rigid part to facilitate further manipulations, such as (e) pouring the balls.

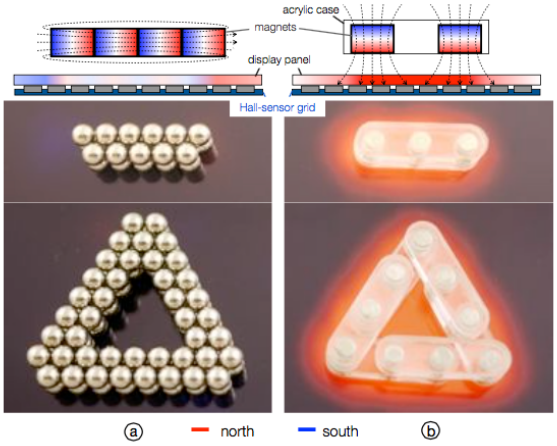


Figure 3. Magnetic fields shaped by two construction methods. (a) Non-uniform magnetic fields caused by attracting the magnets. (b) Uniform magnetic fields shaped by firm casing. A laptop display mounted with an analog Hall-sensor grid visualizes N- and S-polar magnetic field intensity maps in red and blue, respectively.

In the tangible flight simulation (Figure 13c), users grasp his airplane assembly as the controller, tilt it in four directions to steer or pan in the context (Figure 13d), and lift or lower it to change the height of the flight naturally.

A virtual pet application (Figure 15b) demonstrates the enabled touch input capability. Users use touch blocks and non-touch blocks to construct a sleeping cat. Users pat the cat to wake it up, please it by sliding their finger along its body (Figure 15c), and irritate it by pinching its body (Figure 15d). The cat's facial expression changes according to his feelings. While performing multitouch gestures on the cat's body, the users press the non-touch block to prevent the model from moving away.

We introduce a 3D puppetry system that allows users to quickly create 3D animations by performing the motions with their own familiar, rigid toys, props, and puppets. As shown in Figure 1, the puppeteer directly manipulates these objects in front of an inexpensive Kinect depth camera [10]. We

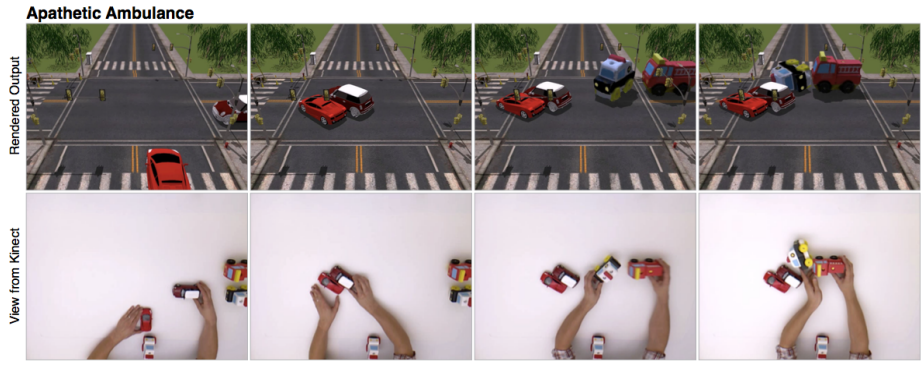


Figure 2. Example frames of our system in action. The bottom row shows the puppeteer manipulating physical puppets. The top row shows the result of our system tracking the puppets in real time and rendering them in a virtual set.

The KinectFusion project [17] allows one to quickly scan a physical environment. We use a similar system, ReconstructMe [13], to convert physical puppets into 3D models.

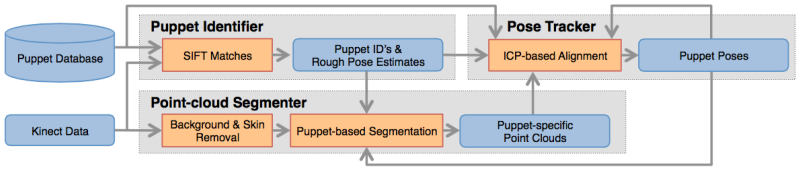


Figure 3. Overview of our system's Capture module. With each frame, the Kinect provides an RGB image and depth map. The puppet identifier compares SIFT features in those data to the features found in a database of image templates to identify puppets and roughly estimate their poses. The RGB and depth information are also combined into point clouds, which are processed to remove the background and the puppeteer's hands, and then matched to stored 3D models using ICP to estimate the puppets' 6D poses.

Figure 6. Steps of point-cloud segmentation. Beginning with the raw point cloud from the Kinect, our segmenter removes the background and the puppeteer's skin, and finally produces separate point clouds associated with each puppet.

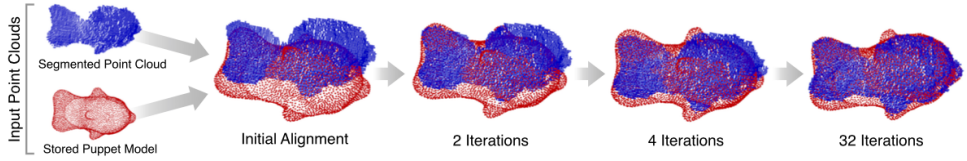
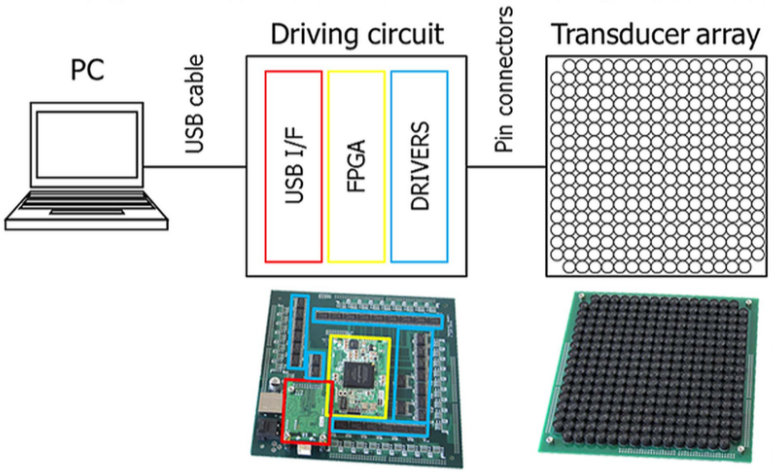
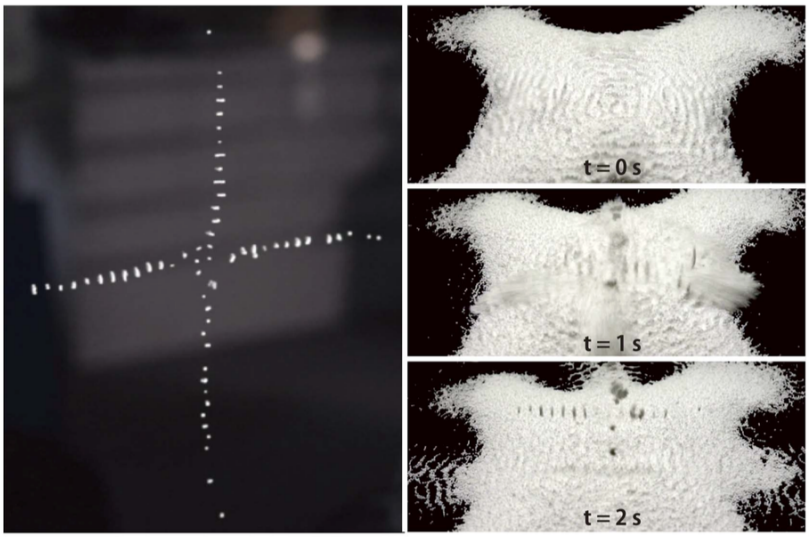


Figure 7. Progression of ICP alignment between a segmented point cloud and a Puppet Database model. Our ICP algorithm begins by transforming the puppet model using its last known pose (if available; otherwise it uses the rough pose estimate from the Puppet Identifier). Then each ICP iteration brings the clouds into closer alignment until the root mean square distance between their points meet an error threshold.

Ultrasonic levitation method has been used to levitate lightweight particles [1], small creatures [2], and water droplets



focal length R was 200 mm. The spatial resolution of the position of the focal point was 0.5 mm, and the refresh rate was 1 kHz.



3. Elephant source programs may not need data structures, because they can refer directly to the past. Thus a program can say that an airline passenger has a reservation if he has made one and hasn't cancelled it.

For example, when a program "promises" someone to do something, it needn't believe (as Searle (1969) suggests it should) that fulfillment of the promise will do its recipient some good. Indeed many programs that promise

Notice that it isn't necessary for most purposes to apply moral terms like *honest*, *obedient* or *faithful* to the program, and we won't in this paper.

2. Questions. The user can question the program, and the program can question the user.

goals. The most important features of this informatic situation are independent of the fact that we are humans. Martians or robots with independent knowledge and goals would also require speech acts, and many of these would have similar characteristics to human speech acts.

Algol 48 programs are organized quite differently from Algol 60 programs. Namely, the changes to variables are sorted by variable rather than sequentially by time. However, by reifying variables, Algol 50 permits writing programs in a way that permits regarding programs in this fragment of Algol 60 as just sugared versions of Algol 50 programs.

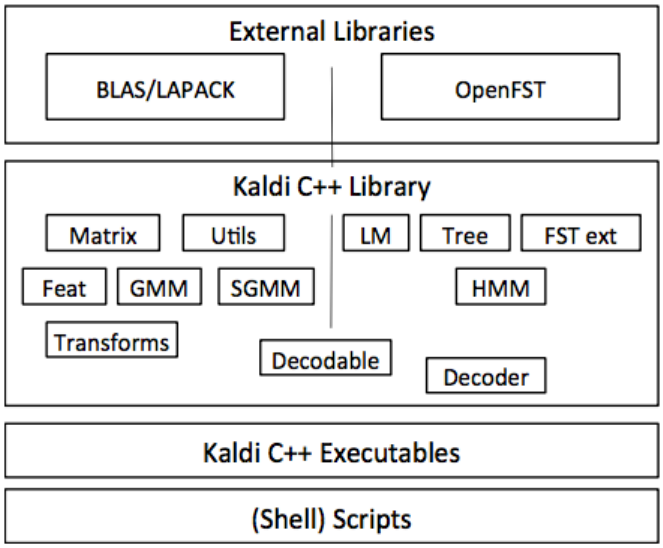


Fig. 1. A simplified view of the different components of Kaldi. The library modules can be grouped into those that depend on linear algebra libraries and those that depend on OpenFst. The *decodable* class bridges these two halves. Modules that are lower down in the schematic depend on one or more modules that are higher up.

To avoid "code rot", We have tried to structure the toolkit in such a way that implementing a new feature will generally involve adding new code and command-line tools rather than modifying existing ones.

VI. Language Modeling

Since Kaldi uses an FST-based framework, it is possible, in principle, to use any language model that can be represented as an FST. We provide tools for converting LMs in the standard ARPA format to FSTs. In our recipes, we have used the IRSTLM toolkit [3] for purposes like LM pruning. For building LMs from raw text, users may use the IRSTLM toolkit, for which we provide installation help, or a more fully-featured toolkit such as SRILM [4].

iVectors which capture both speaker and environment specific information have been shown to be useful for rapid adaptation of the neural network [7, 8, 9]. iVector based adaptation has also been shown to be effective in reverberant environments [10]. In this paper we use this adaptation technique.

Reverberant speech is assumed to be composed of direct-path response, early reflections and late reverberations. Reflections within a delay of 50ms of the direct signal are categorized as early reflections. Late reverberations, which comprise of later reflections, have *reverberation time* from 200 to 1000 ms in typical office environments [1]. Early reflections can be

Decoding the entire 10 minute recording as one segment is not possible due to round-off in the decoder. We segmented the recordings into chunks of 10 seconds long each, shifted by 5 seconds each time. There was no attempt to make the chunk boundaries coincide with silence. We reasoned that if a recording is cut in the middle, only the part of the transcript near the cut point will be affected, so we filtered the transcripts by removing words whose midpoints were within 2.5 seconds of the edge of its chunk of origin, before combining them into a single long transcript.

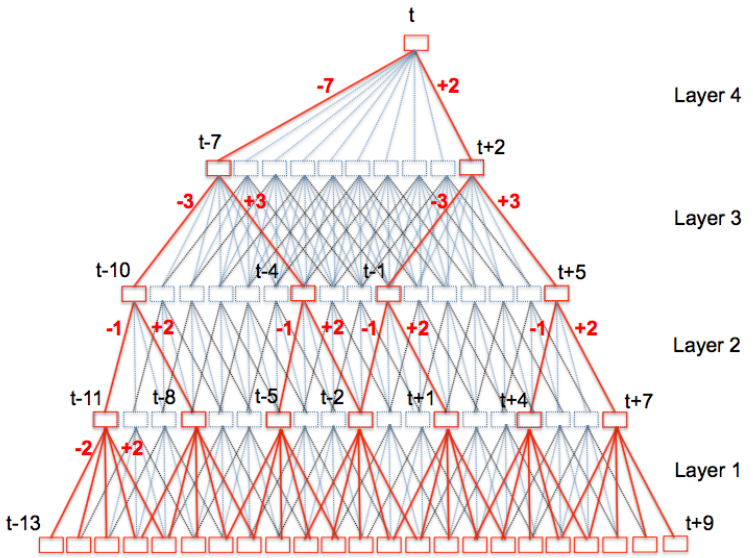


Figure 1: Computation in TDNN with sub-sampling (red) and without sub-sampling (blue+red)

Recent advances in algorithms and computer hardware have made it possible to train neural networks in an end-to-end fashion for tasks that previously required significant human expertise. For example, convolutional neural

While automatic speech recognition has greatly benefited from the introduction of neural networks (Bourlard & Morgan, 1993; Hinton et al., 2012), the networks are at present only a single component in a complex pipeline. As with

The goal of this paper is a system where as much of the speech pipeline as possible is replaced by a single recurrent neural network (RNN) architecture. Although it is

$\mathcal{H}$ is usually an elementwise application of a sigmoid function. However we have found that the Long Short-Term Memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997), which uses purpose-built *memory cells* to store information, is better at finding and exploiting long range context. Fig. 1 illustrates a single LSTM memory cell. For

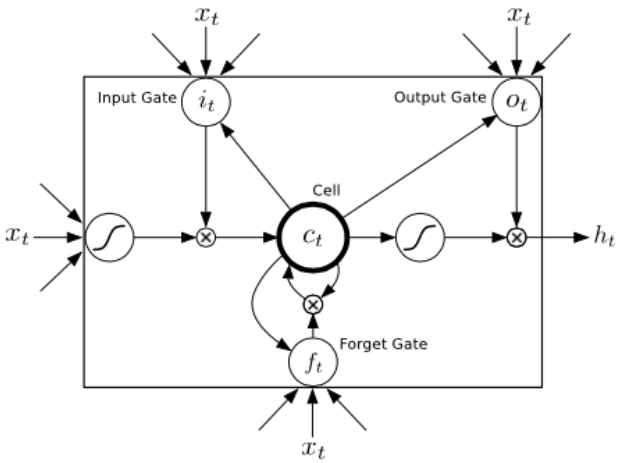


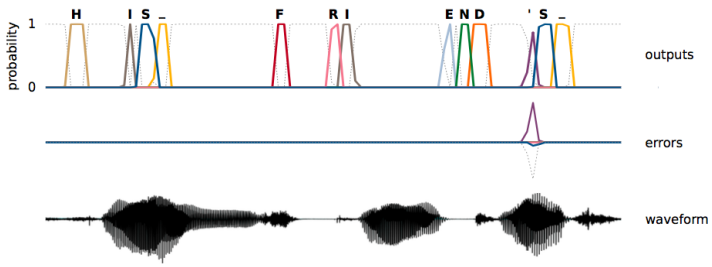
Figure 1. **Long Short-term Memory Cell.**

scriptions matter. Connectionist Temporal Classification (CTC) (Graves, 2012, Chapter 7) is an objective function that allows an RNN to be trained for sequence transcription tasks without requiring any prior alignment between the input and target sequences.

To provide character-level transcriptions, the network must not only learn how to recognise speech sounds, but how to transform them into letters. In other words it must learn how to spell. This is challenging, especially in an orthographically irregular language like English. The following

target: *T. W. A. ALSO PLANS TO HANG ITS BOUTIQUE SHINGLE IN AIRPORTS AT LAMBERT SAINT*

output: *T. W. A. ALSO PLANS TOHING ITS BOOTIK SINGLE IN AIRPORTS AT LAMBERT SAINT*



"Deep learning. How well do you think it would work for your computer vision problem?" Most likely this question has been posed in your group's coffee room. And

Answer: Surprisingly the CNN features on average beat poselets and a deformable part model for the person attributes labelled in the H3D dataset. Wow, how did they do that?! They also work extremely well on the object attribute dataset. Maybe these `OverFeat` features do indeed encode attribute information? (Details in section 3.5.)

- The feature vector is further $L2$ normalized to unit length for all the experiments. We use the 4096 dimensional feature vector in combination with a Support Vector Machine (SVM) to solve different classification tasks (CNN-SVM).

Method	mean Accuracy
HSV [27]	43.0
SIFT internal [27]	55.1
SIFT boundary [27]	32.0
HOG [27]	49.6
HSV+SIFTi+SIFTb+HOG(MKL) [27]	72.8
BOW(4000) [14]	65.5
SPM(4000) [14]	67.4
FLH(100) [14]	72.7
BiCos seg [7]	79.4
Dense HOG+Coding+Pooling[2] w/o seg	76.7
Seg+Dense HOG+Coding+Pooling[2]	80.7
CNN-SVM w/o seg	74.7
CNNaug-SVM w/o seg	**86.8**

Table 4: **Results on the Oxford 102 Flowers dataset.** All the methods use segmentation to subtract the flowers from background unless stated otherwise.

Thus, it can be concluded that from now on, deep learning with CNN has to be considered as the primary candidate in essentially any visual recognition task.

Our object detector is based on a very simple idea: to learn *a separate classifier for each exemplar* in the dataset (see Figure 2). We represent each exemplar using a rigid HOG template [7]. Since we use a linear SVM, each classifier can be interpreted as a learned exemplar-specific HOG weight vector. As a result, instead of a single complex

Figure 4. **Exemplar-SVMs.** A few "train" exemplars with their top detections on the PASCAL VOC test-set. Note that each exemplar's HOG has its own dimensions. Note also how each detector is specific not just to the train's orientation, but even to the type of train.

At test time, each Exemplar-SVM creates detection windows in a sliding-window fashion, but instead of using a standard non-maxima-suppression we use an exemplar co-occurence based mechanism for suppressing redundant responses. For each detection we generate a context feature similar to [3, 9] which pools in the SVM scores of nearby (overlapping) detections and generates the final detection score by a weighted sum of the local SVM score and the context score. Once we obtain the final detection score, we use standard non-maximum suppression to create a final, sparse set of detections per image.

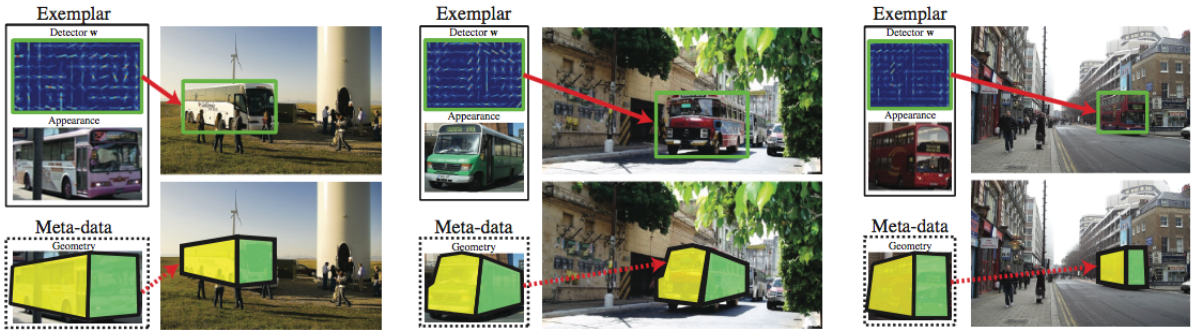
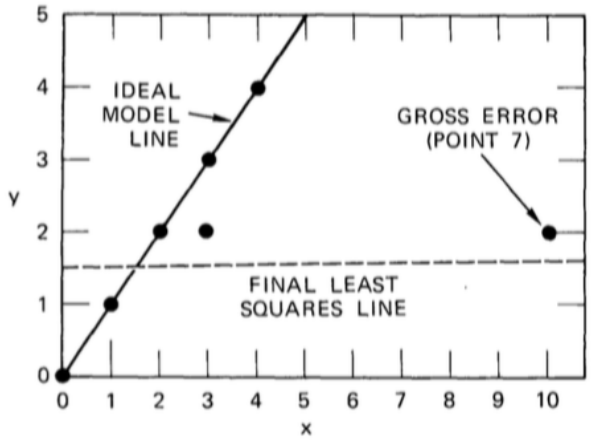


Figure 8. **Qualitative Geometry Transfer.** We transfer geometric labeling from bus exemplars onto corresponding detections.

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, interpretation involves two distinct activities: First, there is the problem of finding the best match between the data and one of the available models (the classification problem); Second, there is the problem of computing the best values for the free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent—a solution to the parameter estimation problem is often required to solve the classification problem.



POINT	x	y
1	0	0
2	1	1
3	2	2
4	3	2
5	3	3
6	4	4
7	10	2

A basic problem in image analysis is establishing a correspondence between the elements of two representations of a given scene. One variation of this problem,

E. A "Real" Location Determination Problem

Cross correlation was used to locate 25 landmarks in an aerial image taken from approximately 4,000 ft with a 6 in. lens. The image was digitized on a grid of 2,000 × 2,000 pixels, which implies a ground resolution of approximately 2 ft per pixel. Three gross errors were

3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors. This paper will discuss fast nearest-neighbor algorithms that can perform this computation rapidly against large databases.

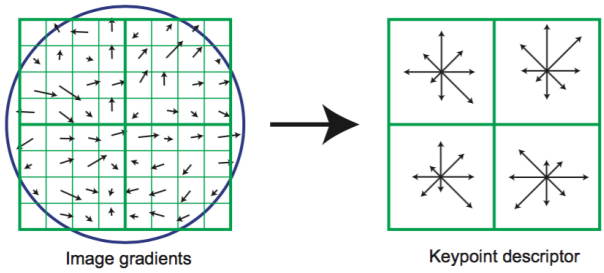


Figure 12: The training images for two objects are shown on the left. These can be recognized in a cluttered image with extensive occlusion, shown in the middle. The results of recognition are shown on the right. A parallelogram is drawn around each recognized object showing the boundaries of the original training image under the affine transformation solved for during recognition. Smaller squares indicate the keypoints that were used for recognition.