The Visual Computing Database: A Platform for Visual Data Processing and Analysis at Internet Scale

1 Introduction

There is strong evidence that a fundamental requirement of the next generation of visual and experiential computing (VEC) applications will be the efficient analysis and manipulation of large repositories of visual data (images, videos, RGBD, point clouds, etc.). For example, highly publicized recent advances in image understanding [24] stem not only from improved deep-learning techniques, but also from the ability to execute these techniques on internet-scale, human-annotated image collections [11, 28, 1, 45]. Analysis of visual data collections is also a major component of other advanced VEC applications such as computational photography, high-level image and video manipulation [19, 26, 25], and 3D scene reconstruction [36] and understanding [14]. Further, with image sensors rapidly becoming ubiquitous in all environments (in public spaces, on vehicles, the body, etc.), we anticipate the emergence of entirely new ecosystems of applications made possible by large-scale mining of streams of visual sensor data. These applications may range from consumer-centric personal assistants, to management of critical infrastructure for transportation and smart cities, and even include data-analysis for fundamental science.

Of course, scaling visual data analysis to facilitate applications that operate on collections such as all public images and videos on Facebook/YouTube (Figure 1-A), all video cameras in a major city (1-C), or petabytes of images in a digital sky survey [42] (1-B), poses significant computer science challenges. First, the sheer size of visual data representations (in comparison to text) results in datasets that far exceed the data-management capabilities of most researchers, application developers, or data analysts. For example, as a service to the community, Yahoo recently computed visual features for the 100 M images in the Flickr Creative Commons dataset [1] (the world’s largest annotated dataset), making the resulting 50 TB of descriptors available to the public. Although 50 TB is a daunting amount of data for most application developers to manage, it is dwarfed by the amount of visual data that is presently available for analysis in the world. For example, nearly three times as many photos are uploaded to Facebook each day! Second, state-of-the-art algorithms for understanding and manipulating large image datasets are computationally expensive and require supercomputing-scale processing. Very few programmers have the capability to develop efficient solutions from scratch at these scales, inhibiting our field’s ability to explore more advanced data-driven VEC applications [1].

In response to these challenges, this project seeks to develop a new parallel computing platform that facilitates the development of applications that require visual data analysis at massive scale. We call the proposed system a Visual Computing Database because it draws ideas from traditional database management systems to more easily and powerfully organize and manage visual data collections using relational models. However, unlike existing systems, it provides a unique set of programming abstractions and system capabilities specific to the needs of future large-scale VEC applications. At the core of our approach is the design and implementation of a new visual data processing and query language that integrates concepts from high-performance functional image processing languages [33, 20] with traditional relational operators, providing the ability to execute sequences of complex image and video analysis operations with exceptionally high efficiency in the database (near data storage). We will implement and evaluate a cloud-based prototype runtime for the visual computing database and release the software as an open source platform to the community. Further, we will work with collaborators from Google to deploy the prototype as a cloud computing service on the Google Cloud Platform, enabling third-party applications (written by other

---

1 The disparity between the large amount of visual data generated in the world and the shortcomings of our ability to extract value from it has led researchers to jokingly call visual data “the dark matter of the internet” [32].
Figure 1: Analysis of large-scale image collections is an increasingly important task in many fields including (A) fundamental computer vision research (extracting knowledge from internet photo and video collections), (B) survey-based astronomy (the LSST will generate 30 TB of image data each night), and (C) security and digital forensics (1000’s of live camera feeds are simultaneously streamed to Manhattan’s video command center). This project seeks to develop a scalable image analysis platform that will make it easy to express visual data queries, and more efficient to execute these queries against large-scale repositories.

analysts or researchers) to use the Visual Computing Database to execute computations at scale on widely used, internet-scale computer vision datasets.

Since in practice visual data analysis workflows will involve more than just data retrieval operations, to ensure a practical system, the Visual Computing Database will be augmented with mechanisms for initiating supercomputing-scale, non-relational computations on visual query results (e.g., via integration with popular large-scale machine learning frameworks for classification and regression) and then ingesting these results back into the database. All together, we aim to empower VEC community with a novel, scalable platform for expressing large-scale visual data analytics.

1.1 Intellectual Merit

The design and implementation of the proposed Visual Computing Database presents many challenges at the intersection of image processing, database system design, and high-performance computing that together form the intellectual merit of this work.

What are good programming abstractions for expressing complex queries on visual data collections?

We seek to integrate high-performance image/video processing kernels as fundamental logic in visual database query operations. Success will require answering questions such as: how should relational data models be extended with first-class support for common visual computing entities such as images and videos, as well as derived entities such as image patches, image descriptors, bounding boxes, or point clouds? How should the scheduling algebra recently developed for high-level image processing languages (such as Halide [33]) be expanded to incorporate the key relational algebra operations (e.g., select, join, group), allowing global optimizations to be made across both domains? How can we incorporate logical predicates such as spatial relations (above, below, nearby, etc.) and temporal relations (before, during, etc.) into the queries?

How should the database system efficiently represent and manage derived and intermediate datasets?

Exploratory data analysis involves sequences of analysis operations (“pipelines”) that generate large intermediate datasets. If materialized in full, these intermediates can often dwarf the original dataset in size. Efficiently managing these intermediates is critical. For example, when should data be recomputed versus stored? And how can we maximize producer-consumer locality (at all levels of the machine) to avoid
What are practical ways to integrate large-scale distributed computation (e.g., for machine learning and analysis) with a visual database system? Modern approaches to image understanding rely heavily on machine learning techniques. Thus, any practical system for visual data analysis must not only support efficient database query operations, but also provide support for seamlessly executing complex analyses on the results of queries. Common operations, such as training deep neural network models, involve expensive computations or large datasets that are often distributed over many machines. To maintain high performance the system must also avoid unnecessary data marshaling between these two systems.

What are hardware platform requirements for large-scale visual data analysis applications? While the focus of this proposal is not to explore specialized hardware for visual data analysis, we intend to analyze the characteristics of large-scale visual computing applications using prototypes running on a variety of platforms. We will build highly tuned implementations for traditional data centers built from multi-node, multi-core CPUs, as well as throughput-computing architectures such as Xeon Phi and GPUs. The results of this analysis will provide guidance for hardware designers seeking to specialize future hardware architectures for this domain, and for system architects who wish to build large data centers.

1.2 Broader Impacts

Large scale data analytics is critical to science, medicine, government, and business. Analyzing video and images is particularly challenging, because of the amount of data and the difficulty of extracting semantic information from pixel data. Image database and analysis systems stand to have significant impact on a variety of domains including:

Fundamental research in computer vision and graphics. We believe our system would enable computer graphics and vision researchers to test their algorithms more easily and quickly than they can do now, hopefully increasing the rate of progress in the fields. The database will also be useful for computational photography researchers seeking to develop algorithms that leverage large image databases for image manipulation tasks.

Scientific instrumentation Many areas of science use instruments that generate large collections of images that need to be processed. Sky surveys like Large Synoptic Survey Telescope (LSST) are anticipated to generate over 30 TB of images a night. Processing and analyzing the images in these sky catalogs will be an important aspect of astronomy research in the coming decade. Similarly, biological medical researchers involved in projects like the Human Brain Project and Human Connectome Project will be generating collections of very large images that need to be processed. More generally, many future scientific instruments will be based on imagers which will generate massive amounts of imagery that needs to be processed and made available to the scientific community.

Security and digital forensics: At the scale of local policing and also the broader scale of international human rights investigations. (Example of organizations/citizens/observers posting videos on-line, and organizations trying to mine through the unstructured collections to establish the location of images as well as the people and objects in them.)

General analytics and decision making. Because cameras are cheap and ubiquitous, image and video processing is becoming widespread and many applications are emerging. One example (see Section[4] example) is understanding traffic patterns in urban environments. By monitoring traffic in real-time, it is possible to improve the efficiency of public and private transportation systems. We believe our system ultimately will have many uses in the commercial world.
A central focus of this project is to develop a working system that allows a large community of users to regularly issue complex queries of large image collections. We will build a scalable visual database system using the Google Cloud Infrastructure, and populate it with millions of images. We will release our software as open source, so others can build upon our work.

Finally, because imaging is so computationally extensive, progress in image analysis has the potential to stimulate the high performance computing industry, and projects of strategic national importance such as the Exascale Computing Initiative. For example, recent breakthroughs in image understanding based on convolutional neural networks and deep belief networks have been quickly put into practice by large companies such as Google, Microsoft, Facebook and Baidu. This in turn has created a large demand for new data centers with significantly increased computational capability, which has benefited HPC vendors such as IBM, Intel and NVIDIA.

2 Background

Languages for high-performance image processing. Recently, systems such as Halide [33] (created by essential personnel Jonathan Ragan-Kelley and Darkroom [20] (created by Co-PI Hanrahan) have demonstrated that image processing pipelines expressed in a high-level functional representation can be compiled to highly optimized implementations for parallel architectures ranging from mobile phones to GPUs to FPGAs. For example, the Halide language has been demonstrated to generate pipelines that outperform hand-tuned implementations by advanced programmers. Halide is now in widespread use in industry and has been used to build state-of-the-art image processing systems including Google’s HDR+ pipeline featured in millions of Android phones (as well as Google Glass) and Google Photos’ AutoEnhance feature, which runs on tens of thousands of data center machines. Key to Halide’s success is the scheduling algebra it defines for concretely implementing pipelines of functional image processing operations. This success mirrors that of the use of relational algebra for query planning and optimization in database systems. In this project we seek to unify these two worlds (image processing pipelines and relational algebra operations) to enable simple, but performant, expression of complex visual data analysis queries. The key to doing this successfully will be tracking and managing intermediate data simultaneously through both the relational and pixel/array-processing worlds, from small blocks of pixels in L1 cache to terabytes of patch features in a distributed file system.

Domain-specific languages and runtimes. In addition to domain-specific frameworks for image processing, the project PI’s and essential personal have a strong track record of developing domain-optimized programming and runtime systems. PI Fatahalian and Co-PI Hanrahan collaborated as part of the research team that created the Brook stream processing language GPUs [4], a system that was influential in generalizing graphics systems to domains beyond rendering (a precursor to industry standard languages such as CUDA and OpenCL). PI Fatahalian and Co-PI Hanrahan also collaborated on the design of the Sequoia programming language [16], a distributed runtime system that served as the precursor to the Legion runtime system [3] used as the primary distribute compute engine for the prototype developed as part of proposed work. Co-PI Hanrahan has been involved in many recent efforts to develop high performance languages and runtimes for domain-optimized systems, leading to the creation of Liszt [13], Riposte (a JIT compiler for R) [38], and Terra [12], a framework for building domain specific languages (that will be used to implement the proposed visual data query language compiler in this project).

Why a new database specific to visual computing? In recent years, the challenge of analyzing increasingly large data volumes has triggered a variety of database design efforts that explore problem-specific data models and new mechanisms for expressing computation inside database engines. While all systems
(including the system we describe in this proposal) are inspired by the same basic tenants (that computation must be moved into the database engine for efficiency, and that highly structured programming models are necessary to simultaneously achieve high performance and application programmer productivity), their design varies greatly based on domain of focus: e.g., graph processing (Neo4j [30], FlockDB [40], Teradata’s SQL-GR [39]), general machine learning (MLBase [23]), or array-based scientific and numerical computation (SciDB [7]). While many of these existing systems provide functionality that is well aligned with the machine learning aspects of visual computing workloads, no existing system provides an ideal platform for executing complex image and video analysis. For example, although SciDB exports a first-class notion of 2D array types that can be used to store images, the database provides no mechanisms for expressing high-performance pixel processing pipelines on these entities. Such pipelines are the fundamental component visual data analysis applications. GIS database systems [15] also fail to meet the needs of visual data analysis applications for similar reasons. (Although, as stated in Section 4, we anticipate 2D and 3D spatial joins, a common operation in GIS databases, will also be an important operation in a visual computing database.)

3 A Visual Data Analysis Scenario

To gain intuition about the needs of visual data analysis workloads, consider a scenario where an analyst seeks to understand non-vehicular traffic in a metropolitan downtown region (e.g., to recommend locations for new bike lanes in the city). At the analyst’s disposal is a large corpus of video recorded over the past year from the city’s many outdoor security and traffic monitoring cameras. The analyst, employing state-of-the-art computer vision techniques [17], might seek to count instances of bicycles on each downtown city block by crafting a bicycle detector using a pipeline of image processing operations as illustrated in Figure 2. Given a video frame, this pipeline identifies image regions (at multiple scales) that are likely to contain objects [41] (a step implemented via low-level image processing filters), and then applies a widely-available bicycle classifier to these regions (evaluation of a deep-neural network followed by a linear SVM classifier) [24, 17]. To generate the desired count of bicycle occurrences per city block, image regions containing bicycles must be joined back to the video streams from which they came and with the city block on which the camera resides. Finally, aggregation within blocks yields the final counts. Although conceptually simple to describe, this content-based visual query pipeline, when executed on a year’s worth of video from hundreds of cameras, involves executing a many complex operations on terabytes of image pixels. More nuanced analyses would require expression of additional logic beyond just selection. For example, questions about the danger to cyclists on particular city blocks might be answered by not only detecting the presence of bicycles and vehicles in an image, but also computing the spatial proximity of their bounding boxes.

Many visual analysis tasks will also necessitate substantial processing on the results of visual queries. Consider an extension of the example above, where the traffic analyst wishes to ask further questions such as what percentage of the cyclist population wears a helmet? As it is unlikely for a detector of such a specific scene concept to be readily available, the analyst might seek to import images of cyclists wearing helmets into the database e.g., by saving the results of a Google Image Search for “cyclist with helmet” then using the annotated images as training data task-specific classifier to be used in a subsequent query. We note that best-practice training methods involve substantial amounts of processing, necessitating efficient distributed processing [27, 34, 9].
4 Proposed Work: A Database for Visual Data Processing and Analysis

Although presented in the context of a very specific data analysis scenario, we believe the image analysis pipeline featured in Section 3 is representative of future every day visual data analysis tasks across a wide variety of disciplines. To meet the needs of these emerging applications, we believe that a platform for visual data analysis at scale must:

- Ease the burden of managing terabyte-to-petabyte scale visual data collections, which includes both original source data and (potentially cached) intermediate collections generated as a result of analysis operations.

- Provide a high level visual data query language that facilitates rapid expression of complex queries and analyses that operate directly on the visual content. This language must emit efficient implementations (near hand-tuned performance) on modern throughput architectures such as multi-core CPUs and GPUs (and afford the possibility of future acceleration by specialized hardware or FPGAs).

- Provide mechanisms to efficiently use the results of visual queries in large-scale parallel computations, in particular machine learning.

We discuss the details of how the proposed Visual Computing database will address these system requirements in the following subsections.

4.1 Modeling Image Collections as a Relational Database

In most visual data analysis scenarios today (like the one described in Section 3), visual data is stored as an ad-hoc collection of image or video files alongside files containing disparate, inconsistently-structured metadata about the visual assets. An analyst seeking to perform a task must collect and organize the files, manually gather and normalize the relevant metadata (e.g., the city block in which each camera is located), and build custom scripts to tie together many stages of complex processing. To execute the computation, terabytes of data must be ingested into a distributed or network file system (e.g., NFS or HDFS), computation manually distributed over a cluster, and storage of intermediate values manually considered and managed.

In response to this complexity, a core idea of this proposal is that large collections of images are better represented, managed, and processed using a relational database model than as a tree of files in a traditional file system. The language of relations and relational algebra is a natural fit for operating on collections of images and their associated metadata. For example, images often are associated with the GPS position where they were taken, camera settings from the shot, and additional annotations (e.g., “contains a cyclist”) that might have been generated from prior analyses and provide information about their contents. Although the advantages of a relational data representation, such as “physical independence”, structured data management and administration, efficient and flexible computation from simple declarative queries, implicit parallelism and distributed execution, are well established in many other domains, image collections are still often managed as sets of raw files because traditional database management systems are not optimized for either the computation or storage requirements of large-scale visual data.

Our proposed visual computing database seeks to provide applications many of the benefits of traditional database management systems while also delivering visual data storage and computation efficiency matching or exceeding the hand-designed visual data processing systems today. A key aspect of achieving this goal is the addition of first-class database support for visual data types and pixel-level visual data processing on these types. For example, in addition to traditional database column types such as numbers, strings,
dates, the proposed database will also feature support for images, videos, and RGBD data. Since processing pixel data presents a substantially different workload than numbers and text, the proposed system will separate the storage of metadata about images and videos (keeping it resident in traditional relational data structures) from the storage of pixel data. We intend to start by extending the Postgres RDBMS for metadata storage and queries, and coupling it to a distributed object store (such as Amazon S3 or OpenStack Swift) for storing pixel data, and the Legion runtime for distributed in-memory caching and computation. While the relational layer will continue to manage and arbitrate access to pixel-level data, this data will be located in a subsystem that is optimized for inexpensive and scalable storage of large binary blobs (the object store). Queries on database fields corresponding to pixel data will include complex pixel-level computations that transform images to other images (e.g., reconstruction, filtering, and resampling operations) or into popular image descriptors (e.g., SIFT, HOG, “deep features”).

First-class support for domain-specific visual-data types, as opposed to more general array-based types (e.g., SciDB) affords improved opportunities for data compression. Our storage system will rely on domain-specific image and video compression techniques wherever possible to reduce storage footprint and bandwidth requirements. Images and videos also carry explicit metadata describing the asset’s visual interpretation, such as color space and gamma curves with which it is encoded. Failing to track and account for different encodings is a major source of image processing errors that a structured representation can help eliminate.

4.2 Defining a Visual Query Language

A second key component of our proposed work is the design of a new visual query language used to retrieve information from the visual data stores described in Section 4.1. Visual data queries require efficient first-class pixel processing operations as a core part of the query language. Like the Halide language, the query language will model high performance pixel manipulation as pipelines of pure user-defined functions. However, to support expression of complex selection logic, the language augments existing Halide constructs with key relational algebra operations as well as logical predicates for both spatial and temporal relations. Whereas Halide programs operate only on a single data type—functions defined on multi-dimensional
regular grids—the query pipelines expressed in this new query language operate on an array of object types generated as a result of query processing, including collections, image patches, and feature descriptors. To establish intuition about the type of queries this language must support, consider the pseudocode for the previously described bicycle traffic analysis task shown in the bottom-left of Figure 2. It proceeds as follows:

- The pipeline begins by selecting relevant videos from the database, and extracting patches by building pyramids of each video frame. This is (1) a traditional relational selection, based on metadata about the video, followed by (2) a relational join of video frames with the set of all patch locations to produce the set of all patches at all pyramid scales for all videos, and finally (3) image processing of the video pixels to produce the pyramid patch pixels.

- Next, the pipeline performs a selective search process on those patches’ pixel data to filter out patches less likely to contain objects of interest [41]. This is a combination of image processing on patch pixels to compute an “objectness” score, followed by relational selection of patches based on a score threshold.

- The pipeline then warps the selected patches to a standard input size for a convolutional neural network, applies the network to each patch to compute mid-level features, and applies a linear classifier to those mid-level features. Each of these operations is a traditional image processing operation on an individual patch; together, they form a traditional image processing pipeline.

- The pipeline next uses the classifier response to filter the patch set to only those containing bicycles, and joins back to the metadata of the original videos from which the patches came. This is a relational selection of patches based on classifier response, followed by a relational join of patches to video metadata.

- Finally, the pipeline geospatially joins bike matches to the city blocks where they were filmed, and aggregates the detection counts by block ID. This is a composition of relational and spatial query operations.

We now sketch additional examples of visual queries that we will consider during the design of the visual query language. Below we use a SQL-like syntax to depict the queries in order to highlight the relational aspects of these operations. However, we anticipate that a functional language syntax will be a more suitable final form of the visual query language.

**Selecting “similar” images in feature space.** One common data analysis operation is to query a database for “similar” images as defined in a desired feature space. For example, the following query requests the top 10 neighbors in feature space for any images containing a cat:

```sql
select * from images where catDetector(image) > threshold
```

This query first selects images containing detections of an object category (cats), using a predefined pipeline (catDetector) which could be internally similar to the bike detection pipeline described previously. The query then constructs the cross product of all images containing cats with all other images, using a join. For each of these images it then projects them into a feature space using myFeatureTransform, a complete image processing pipeline which takes image pixels and returns a feature vector description of the image. (For
example, this pipeline might evaluate the primary layers of a deep convolutional neural network.) Finally, the query sorts the image pairs by their distance in feature space and selects the top 10.

**Selecting images based on spatial relationships between detected objects.** Another common visual analysis technique looks for spatial relationships between objects in an image. We could find all images where a helmet appears above a bicycle with the following query:

```sql
helmets = select bbox, imageID from pyramid(images) where helmetDetector(pixels) > threshold
bikes = select bbox, imageID from pyramid(images) where bikeDetector(pixels) > threshold
select b.imageID in helmets h, bikes b where (h.bbox above b.bbox) and (h.imageID = b.imageID)
```

This query first finds the bounding boxes and image IDs of pyramid patches where either the helmet detector or the bike detector fires. These detectors have been separately defined as pipelines of image- and feature-space computations starting from image patch pixels.

**Iteratively bootstrapping a classifier based on strong detections.** Finally, our query system should also support using visual-relational queries coupled to non-relational computations. One example in contemporary visual analysis which requires non-relational computation for machine learning is using iterative semi-supervised learning to refine a classifier with new data [5]. To refine a car detector from a new batch of unlabeled images, we could run a query to drive a non-relational training algorithm:

```sql
newImages = select image from images where dateAdded > lastTraining
patches = select pyramid(image) for image in newImages
interestingPatches = selectiveSearch(patches)
strongDetections = select patch from interestingPatches where carDetector(patch) > threshold

while (!converged) {
    // iterative map–reduce over query result set:
    map (strongDetections) {
        // compute error...
    }
    reduce {
        // update detector...
    }
}
```

In this example, we first run a complex query to find strong image patches to add to the training set, and then consume the results of this query in a non-relational iterative map-reduce over the result set to update the classifier with the new examples. The query first selects new images which have been added since the last training. It then builds a pyramid of patches over these new images and performs selective search to initially prune the set, as in our other detection pipelines. Next it runs the existing car detector over each interesting patch, and selects patches where the detections are above a high threshold (strong matches). Finally, this result collection is fed through the non-relational distributed training process to update the detector with the new strong examples.

### 4.3 Scheduling Visual Data Analysis Queries

To execute the proposed visual data query language efficiently across large image collections, we must develop a query compiler that unifies techniques for efficiently scheduling functional image processing with methods for implementing relational algebra operations. A naive approach, starting from a generic relational
domain order

Figure 3: (top) Execution strategies for image processing computations expressible by Halide’s schedule algebra. These choices decompose the space of pixel operations in different ways to control the order and granularity of intermediate results between stages. Even simple pipelines exhibit a rich space of scheduling choices, each expressing its own trade-off between parallelism, locality, and redundant recomputation of shared values, determined by how intermediate results are handled. (bottom) Relational algebra operations and data. Traditional relational query planning deals only with these operations over relations. The key challenge in our visual query language will be unifying the data, computation, and scheduling models for these two worlds.

database system, would model pixel operations as black-box external functions to be run at the leaves of the relational operations—e.g., a user-defined function applied by the database independently to individual blobs of pixel data. This is the approach taken by most traditional database systems when confronted with a need for new per-element computations. However, this solution is insufficient to efficiently implement our visual query language since queries may involve interleaved image-processing and relational operations (recall the queries from Section 4.2).

In contrast to the black-box external function approach, SciDB’s “computational database” model instead tries to extend relational algebra with numerical array processing operations [7]. However, for visual analysis tasks, we believe the critical bottleneck is in pixel-level processing, not relational operations, and that space of transformations necessary to express state-of-the-art image processing operations is too large and complex to easily graft into a primarily relational processing system. We therefore propose to take the opposite approach: we will start from the functional Halide [33] language, which is already capable of expressing highly optimized, state-of-the-art image processing pipelines and extend it with relational concepts.

The fundamental challenge of course, is defining a new scheduling algebra that admits composition of scheduling transformations over Halide functions on regular, grid-structured numerical data with relational algebra operations on sets and relational data types (Figure 3). Efficient scheduling was the singular challenge in the design and implementation of the Halide system (without it Halide programs make poor use of throughput-processing resources), and in large-scale image analysis the need for efficient scheduling is even greater. The reason for this is that complex visual data analysis queries generate intermediate results are both large—often much larger than the original data—and frequently very expensive to compute. Because of this, there is a tension between saving intermediate results for potential reuse in subsequent analyses and...
dynamically recomputing them to save storage space. In addition, there is tension between interleaving the computation and consumption of intermediate values at very fine granularity (to exploit producer-consumer locality so as to keep values resident in cache, in memory in a single node, or perhaps across all nodes in a datacenter), and aggregating coarse-grained tasks and values to preserve coherent batch execution. To fully comprehend and globally optimize intermediate computations and management of derived data in these queries, scheduling operations for relational and image processing computations and data must be unified, affording the flexibility to optimize queries across the boundary between relational and pixel/numerical operations. A unified treatment of scheduling will allow optimization decisions to be applied at all system scales, ranging from orchestration of data in the L1 cache of a single core (where Halide focuses now) to the potential materialization of petabytes of storage across a whole cluster.

4.4 Integrating Non-relational Computation for Machine Learning

Applications built on relational database systems often require additional logic outside of the relational query language. A common case we expect to see in visual analysis is training an advanced machine learning system, like a deep neural network, on the results of a query (e.g., training a bike helmet detector as discussed in our urban planning example). Some regression and classification algorithms have been expressed in relational systems [21], but state-of-the-art neural network training using back-propagation and stochastic gradient descent [35] is not natural to implement within a relational system. Furthermore, real users already rely on an ever-expanding wealth of existing libraries, built in traditional languages like C and CUDA, for these and other operations. We therefore cannot limit our system to only in-database operations, but must support integration of non-relational computations over the query result sets.

In traditional database systems this sort of non-relational computation is expressed in general-purpose programming languages, and tied to the database by consuming the results of queries using a database client library. Most often, this client library materializes a complete copy of the result data and streams it across the network to a single client machine. When consuming query results in non-relational application logic, the defining characteristics of visual data analysis are the sheer size of the result data and the amount of computation required to process it. We therefore need to provide facilities to push even non-relational computations, outside of our query language, towards the original and derived data, and to distribute this work across a cluster. Our system will support tight integration with downstream non-relational computation, much like the SparkSQL system exposes results of relational queries as RDDs for further non-relational processing with Spark, in-place on the cluster [44, 43].

5 Open-Source Prototype Implementation

We will implement the ideas described in the prior subsections as a open-source visual database platform. The proposed database will use OpenStack’s Object Store [31] as a durable backing store for database contents, enabling deployment of the database runtime on both private clusters and on commercial cloud platforms. We will develop a compiler for the proposed visual data processing/query language in Terra/Lua [12], leveraging Halide [33] (for image processing kernels) and the Legion parallel runtime system [3] as compilation targets.

We have selected Legion as a low-level parallel runtime environment as it affords the potential for highly efficient parallel implementation across a variety of machine architectures. First, Legion’s logical region abstraction, which encapsulates in-memory collections using a relational model of data, is well aligned with the concept of query result sets in the proposed visual query language. Second, the Legion runtime system
provides a distributed scheduler for launching and managing parallel computations across a heterogeneous collection of processing elements, including multiple machines, multi-core CPUs, and also GPUs. (We anticipate platforms featuring high-density compute capability via GPUs and Xeon Phi accelerators will be attractive execution platforms for visual data analytics.) Last, although Legion provides the service of managing communication and parallel execution in a heterogeneous distributed system, it leaves full responsibility for determining how database operations should be decomposed and orchestrated to our compiler and runtime system. Thus, the database will be able to make efficient scheduling decisions based on high-level knowledge of query structure.

Our implementation will also use the Legion runtime to provide applications with simple mechanisms to launch large distributed computations on the results of database queries (Section 4.4). For example, our initial prototype will provide applications the ability to perform distributed training of deep neural networks using Caffe [22].

Deployment as a Cloud Service

To aggressively exercise our proposed designs, as well as to make the system available to the broader community at scale, in the second half of the project we will work with collaborators at Google (see attached letter of support from Solomon Boulos, with whom both PIs have collaborated successfully in the past—five joint publications since 2009) to deploy the Visual Computing Database runtime as a service on the Google Cloud Platform. This deployment will be run by the PIs (it is not an official Google service) but its deployment will mimic how existing services such as Google Cloud SQL and Datastore are available to applications running on the Google App Engine [18].

To alleviate the burden of massive data ingest for users of the service, we will pre-populate the database service with several large-scale datasets widely used in computer vision research, including: the full ImageNet dataset [11], the Flickr Creative Commons Dataset [1], and the Microsoft COCO Dataset [28]. In addition to original source image and video data, we will also store materialized copies of commonly-used derived quantities such as image features and object bounding boxes.

The deployment of the Visual Computing Database as a service (with common datasets held in a centralized location) provides new opportunities for the visual computing community to efficiently collaborate and share processing and results. For example, intermediate values generated by one research group (that may be expensive to compute) can be cached by the database and made available to other groups in need of similar computations. Further, a common database language, data model, and execution platform will facilitate code sharing/reuse and simplify evaluation of new new analysis techniques. (Researchers can easily run other researchers code!)

6 System Evaluation

No common benchmarks for visual data analytics currently exist, so to evaluate the proposed system we plan to implement a suite of visual query benchmarks in the new visual query language. We will draw our initial benchmarks from sources such as data analysis pipelines commonly used in the computer vision community [17, 41, 14, 5]. We will also seek to include benchmarks representative of queries needed by other domains such as urban traffic planning, digital forensics, and large-survey astronomy.

Using this benchmark suite, we will evaluate the scalability of the prototype database implementation running on the Google Cloud Platform under a variety of system operating conditions. We will measure scalability of queries with increasing platform size (number of nodes and cores per node), under varying database size (number of images/videos, number of features), and varying query complexity (complexity of feature
computation, number of joins, etc.).

The PIs also have access to a research cluster at Carnegie Mellon with compute nodes containing both GPU and Xeon Phi accelerators. Using this cluster we will evaluate the performance of different CPU architectures on our benchmark visual data analysis workloads. This evaluation will also study platform efficiency under varying database operating conditions.

Finally, to better understand visual analytics workloads (beyond merely those in our benchmark suite) we will develop a website that tracks usage statistics of queries run against the Google Cloud Platform database service. For example, the website will post statistics of queries (images touched, cycles used, joins performed, etc.) against the database for the general public.
References


