Astron. Nachr. / AN **328**, No. 8, 852 – 857 (2007) / **DOI** 10.1002/asna.200710817

Multidimensional indexing tools for the virtual observatory

I. Csabai[1,⋆], **L. Dobos**[1], **M. Trencséni**[1], **G. Herczegh**[1], **P. Józsa**[1], **N. Purger**[1], **T. Budavári**[2], and **A.S. Szalay**[2]

[1] Dept. of Physics of Complex Systems, Eötvös University, H-1518 Budapest, P.O. Box 32, Hungary
[2] Dept. of Physics and Astronomy, The Johns Hopkins University, Baltimore, MD, USA

Received 2007 Mar 18, accepted 2007 May 24
Published online 2007 Sep 18

Key words astronomical databases: miscellaneous – methods: data analysis

The last decade has seen a dramatic change in the way astronomy is carried out. The dawn of the the new microelectronic devices, like CCDs has dramatically extended the amount of observed data. Large, in some cases all sky surveys emerged in almost all the wavelength ranges of the observable spectrum of electromagnetic waves. This large amount of data has to be organized, published electronically and a new style of data retrieval is essential to exploit all the hidden information in the multiwavelength data. Many statistical algorithms required for these tasks run reasonably fast when using small sets of in-memory data, but take noticeable performance hits when operating on large databases that do not fit into memory. We utilize new software technologies to develop and evaluate fast multidimensional indexing schemes that inherently follow the underlying, highly non-uniform distribution of the data: they are layered uniform indices, hierarchical binary space partitioning, and sampled flat Voronoi tessellation of the data. These techniques can dramatically speed up operations such as finding similar objects by example, classifying objects or comparing extensive simulation sets with observations.

1 Introduction

The fast development of technology, especially in computer hardware and microelectronics devices is revolutionizing most of the natural sciences with a sudden increase in the amount of measurement and simulation data (Szalay & Gray 2006). In astronomy and cosmology the two major factors of revolution were the appearance of CCD chips and fast computers with cheap, large storage and processing capacity. Ten or twenty years ago cosmology was a theoretical science: there were many models, but there was no data to decide which one is valid. The major cosmological constants were determined only up to a factor of two. A decade later most of them are known with a few percent accuracy. As old questions are resolved, new ones, such as the nature of dark matter and dark energy take their place. The first step in approaching these problems involves large scientific surveys which accumulate observation data on the terabyte scale.

In the last few years several large surveys have collected multi-spectral observational data from radio through to X-ray frequencies, and most of the data is accessible through the internet. Astronomers often have to combine the information from several of these archives to get all the multiwavelength or temporal properties of the objects they study. The mission of the International Virtual Observatory Alliance (IVOA; Quinn et al. 2004) is to facilitate the international coordination and collaboration necessary for the development and deployment of the tools, systems and organizational structures necessary to enable the international utilization of astronomical archives as an integrated and interoperating virtual observatory. The IVOA coordinates the work of more than a dozen national Virtual Observatories (VO) and working groups and several new scientific finding have already been published through VO-based research.

Astronomers have traditionally used large, flat files (e.g. in FITS format) to store data and IDL, Fortran or C programs to analyze them. The architecture of current file systems makes this strategy prohibitive when the size of the data exceeds the gigabyte regime. To overcome this problem the scientific communities have started to use modern database technology, mainly the standard relational database products. One of the earliest and most elaborated examples is SkyServer (http://skyserver.sdss.org) for the Sloan Digital Sky Survey (SDSS) (Stoughton et al. 2002). SDSS creates a detailed digital map of a large portion of our Universe and store several terabytes of data in a publicly accessible archive.

Relational database management systems which are robust enough for use with serious science archives are the same products used by commercial companies. They were mainly designed to meet commercial requirements, not the scientific community's. The current situation with most of the existing scientific databases, partly even with SkyServer is that they are inherently static. They store and provide the data, but most of the scientific data mining is done 'offline'. Also because SQL, the language of database systems, is not flexible enough for complex analysis, scientists only

⋆ Corresponding author: csabai@complex.elte.hu

use it to filter data, and in later phases still use Fortran and C programs for complex analysis. This process usually involves unnecessary conversion of data between various representation formats since code is not running in the same place where data is stored. Nonetheless, the database is still very useful; one can make quite complicated queries to efficiently filter out the needed objects from a huge archive. Based on the new development of the database and software technologies like the so called Web Services, our approach is to move the data intensive processing to the place where the data resides.

A crucial difference compared to business and textual data – the standard target of current database systems – is that most science data is multidimensional and continuous (coordinates, time, brightness of a star, size of a galaxy) in nature. In a geometrical interpretation, the observed quantities are just points in a multidimensional space, scientific theories and models are hyper surfaces establishing relations between them or separating them into various classes. Observed spectral energy distributions can be represented in an even higher dimensional space. To handle efficiently large multidimensional databases, we have to develop efficient indexing techniques. Partitioning and indexing of multidimensional spaces is not a new topic, Oct-tree, R-tree, SS-tree, SR-tree, X-tree, TV-tree, Pyramid-tree and K-d tree (see Samet 2006 for an overview) are just a few of the existing examples. Traditionally, these algorithms are designed and implemented as memory algorithms; it is not trivial to convert them for use with astronomical databases. Also, an algorithm that is optimal as a memory routine can give inferior performance compared to a theoretically suboptimal one when implemented in a database server (Gray et. al. 2004).

The relation between data and analysis is an active, iterative process, astronomers need to intensively interact with the data to find hidden relations. They usually plot data on graphs to visually help this process, but when the number of data points exceeds the limits of the computer's memory or the dimensionality (independent variables) is high, this task becomes cumbersome. Visualization should directly be linked with the database, and this link should be two-way: database indices should speed up visualization and the visual interface should transform visual manipulations into database requests. Geo-spatial tools like Google Earth are good examples for interfaces that dynamically change the level of details (LOD) by getting additional data from Internet based map servers.

Although we use the same data as many of us use in standard astronomical studies, in this paper we take a little bit different perspective from the viewpoint of database researchers. Without exploring this perspective one cannot handle efficiently the large data sets we have already, and the upcoming new surveys like Pan-STARRS or LSST multiplies the magnitude of the problem we face. Based on modern programming technologies and computer geometry we have developed various tools for the VO, to efficiently search, analyze and visualize multidimensional magnitude data and spectral energy distributions.

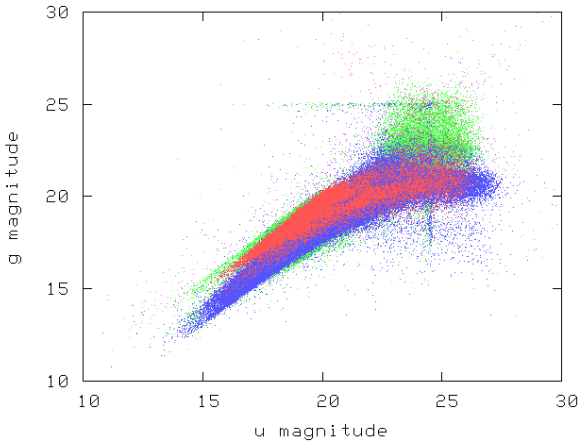

Fig. 1 (online colour at: www.an-journal.org) A 2D projection of the 5 dimensional magnitude space of SDSS. Each dot corresponds to a celestial object; green, blue and red dots are stars, galaxies and quasars respectively. The distribution is highly inhomogeneous and there are outliers.

In Sect. 2 we will introduce a geometric interpretation of the magnitude space, which is a prototype of multiparameter astronomical datasest. Section 3 briefly introduces search trees, indexing techniques and their use during query evaluation. Section 4 provides some applications of the technology and Sect. 5 concludes the paper.

2 The magnitude space

As of early 2007, the current data release (DR5) of SDSS has measured more than 300 parameters for about 270M objects. Among these parameters, beyond the celestial coordinates of the objects, the most important ones are the magnitude values in in five optical color bands: u, g, r, i and z (Fukugita et al. 1996). For objects where spectral observations were not possible, these values carry the most information on the physical properties, like distance, classification, age, temperature, etc. For an illustration let us show a typical example of the projection of the magnitude space.

For multidimensional indexing it is important to notice in Fig. 1, that data points do not fill the parameter space uniformly. There are correlations, points are clustered, they lie along (hyper)surfaces or subspaces. The correlations are sometimes exactly what we are looking for; these are relations between the variables, an expression of the underlying laws. On the other hand this uneven distribution is one of the major problems we face when indexing the data to enable efficient searches. Simple binning or quad tree like structures cannot be used in themselves efficiently; the spatial partitioning must follow the structure of the data. Also note that there are outliers, which are usually results of some measurement or calibration problem, but they can be also

rare, interesting objects. These large variations in the density call for adaptive binning.

The color of points in Fig. 1. corresponds to the so called spectral type of the objects (star, galaxy or quasar). This information is available for less than 1% of the objects in the SDSS database due to observational constraints, but classification of all objects is a crucial task for astronomy. A typical query classifies objects based on their colors, for example separates quasars from other types. To do this, one should identify a few quasars with other measurements (the training set) and then draw a surface in 5 dimensions (ie. in the magnitude space) that best differentiates them from other objects.

Automatic clustering, finding similar objects with drawing a convex hull around the training set or finding nearest neighbors in the magnitude space are a few other typical problems astronomers need to solve, and where a multidimensional geometrical approach can help a lot.

3 Indexing the multidimensional space

Current database management systems store the data on hard disks of computers. The problem is, that the access to the disks is sequential. This means that the multidimensional data should be organized, or *mapped* into one dimension; this mapping is called indexing. We have implemented several spatial partitioning schemes: layered uniform indices, k-d trees (Bentley 1975) and Voronoi tessellation (Aurenhammer 1991) using native SQL and the .NET Common Language Runtime (CLR) features of MS SQL Server 2005. We discuss only k-d trees here.

3.1 K-d trees

To index the multidimensional space we have to do two things: first partition the space into small parts and than order them into some structure that can be handled efficiently. The small partitions will hold the data, their size can be optimized depending on memory and processor speed, and disk buffer sizes. Tree structures provide a very fast and efficient way of searching. In the most popular quad-tree like structures, on each level of the tree, the space is split into two along each axes. The name *quad* comes from the $d = 2$ dimensional version, where accordingly we split every partition into $2^2 = 4$ parts. Searching a single object on average proportional to $\log_{2^d}(N)$ steps compared to the N steps needed to scan through the N individual objects without indexing. With growing dimensions, a problem arises with quad-trees, since the number of the nodes of the tree goes with $(2^d)^l$, where l is the number of levels. In the 5 dimensional magnitude space this means that after 4 cuts along each axes, we partition the space into 1 million bins, each containing a few hundred objects, optimal for the disk buffer size, but the resolution of the partitioning remains very rough, around 1 magnitude. We will see in the following, that this rough resolution makes this index almost useless. One problem is, that as we have shown in Fig. 1, the distribution of the points in the space is highly non-uniform, so most of the final partition boxes will be almost empty, and a small portion of them will hold most of the objects. We have to chose an indexing scheme which is both adaptive, and keep the number of partitions limited.

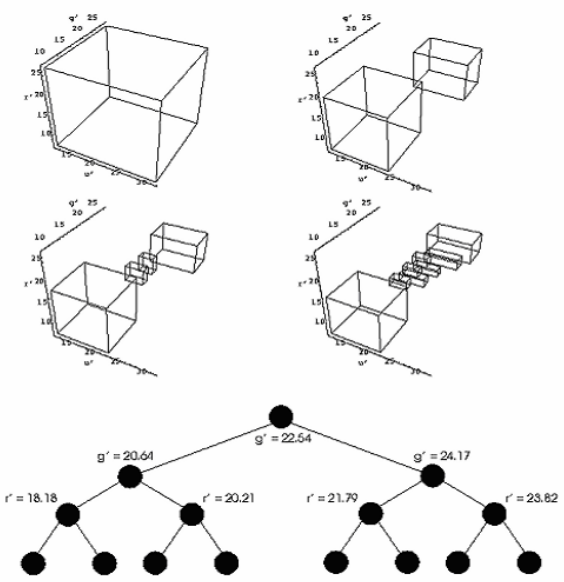

Fig. 2 Hierarchical partitioning of the u, g, r magnitude space. The lower part shows the logical structure of the k-d tree with the values of the cuts, and the upper part illustrates the bounding boxes of the partitions for the corresponding first 4 levels of the tree.

In multidimensional spaces the advantage of k-d tree over most other tree structures is that it applies just one cut at each level. Usually the axis with the largest variance is chosen, and the partition is split into two, in a way that both *child*-nodes contain half-half of the points. In the end, this scheme results a so called balanced tree, where each node contains the same number if objects. The following figure illustrates the scheme for the first few levels of a 3 magnitude space.

Beyond the position of the cutting planes, we calculate and store the bounding boxes of each partition, ie. the minimal hyper rectangle that contains all the points corresponding to the given node. These boxes have a very important role during the evaluation of queries.

3.2 Geometric queries

As an illustration, let us show a real-life query with a complex multidimensional search criterion. It was selected from the more than 12 million user queries of the 2006 May log of the public SkyServer archive.

```
SELECT run, camCol, rerun, field,objID,
b.ra as input_ra, b.dec as input_dec,
g.ra as sloan_ra, g.dec as sloan_dec
FROM   WHERE  and (petroMag_r -
extinction_r < (13.1 + (7/3) * (dered_g
```


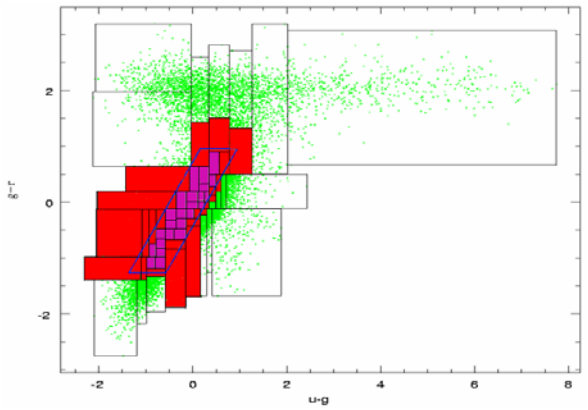

Fig. 3 (online colour at: www.an-journal.org) The demonstration of the evaluation of a query polyhedron (the parallelogram) in 2 dimensions. Green dots are data points, empty cells are outside, purple cells are inside the query polyhedron. The partially covered red cells have to be followed to lower levels of the tree, or at the lower level, database server has to perform detailed SQL search for them.

```
 - dered_r) + 4 * (dered_r - dered_i) -
4 * 0.18) ) and ( (dered_r - dered_i -
(dered_g - dered_r)/4 - 0.18)<0.2) and
( (dered_r - dered_i - (dered_g -
dered_r)/4 - 0.18) > -0.2) and (
(petroMag_r - extinction_r + 2.5 *
LOG10(2 * 3.1415 * petroR50_r *
petroR50_r)) < 24.2) ) or ( (petroMag_r
- extinction_r < 19.5) and ( (dered_r -
dered_i - (dered_g - dered_r)/4 - 0.18)
> (0.45 - 4 * (dered_g - dered_r))) and
( (dered_g - dered_r) > (1.35 + 0.25 *
(dered_r - dered_i)))) and ((petroMag_r
- extinction_r + 2.5*LOG10(2*3.1415*
petroR50_r * petroR50_r) ) < 23.3 ))
```

This typical astronomical question, in the geometrical interpretation translates to a region of the space constrained by a hyper surface. Most surfaces can be well approximated with hyperplanes; the regions with multidimensional polyhedra. When the query is evaluated, the points inside this query polyhedron should be returned. A hierarchical index can substantially speed up the evaluation of the queries. The process is started at the root of the tree. First we decide if the query polyhedron intersect with the node's bounding box. If not, we are done, since the query is not satisfied for any of the objects. Also if the query polyhedron fully contains the bounding box, the evaluation process is terminated for the current node, and all the objects are marked below this node as satisfying the query. We have to continue to the child nodes, and repeat the same process, only if the query polyedron and the bounding box are partially intersecting.

When we reach the lower level of the tree, the so called leaf nodes, objects in partially intersected cells has to be individually tested, if they satisfy the query or not. With the help of the above mentioned acception/rejection process at

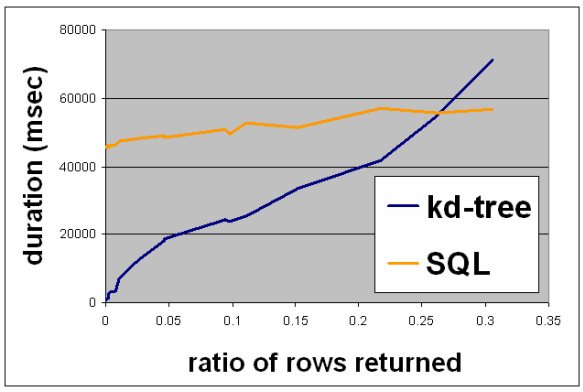

Fig. 4 (online colour at: www.an-journal.org) Performance evaluation of the k-d tree index. For typical queries the number of the points satisfying the query is a small fraction of the whole dataset. In this case k-d tree index can speed up the query by orders of magnitudes.

the bounding box level, the number of points where direct query evaluation is needed is much smaller than without indexing. We have found that if the ratio of the returned vs. total number of rows is below 0.25, kd-trees can outperform simple SQL queries by orders of magnitudes.

Beyond fast query evaluation k-d trees can be used efficiently for outlier detection (Chaudhary et al. 2002), pattern search (Kubica et al. 2005) and nearest neighbor computation (Beis & Loewe 1997), as we will show in example applications.

4 Application for photometric redshift estimation and similar spectrum search

We have created several stored procedures to exploit the capabilities of the spatial indices. They will be published as extensions to SDSS SkyServer and will be included in the next release of our Spectrum Services for the Virtual Observatory tool (Dobos et al. 2004). Using them we can quickly evaluate query polyhedron, find nearest neighbors or similar objects based on a training set of examples. We can also detect outliers based on the volume of the spatial bins. Using the fast efficient indexing we can visualize more than 200 million points adaptively fetching the requested ones from the databases while zooming in and out. We outline two case studies below.

4.1 Photometric redshift estimation with k-d tree nearest neighbor search

To study the large scale structure of the Universe, estimate cosmological parameters from galaxy distribution or for almost all other extragalactic investigations, the measurement of the 3 spatial coordinates of the galaxies are required. While the exact values of the celestial coordinates are easy to get, to measure the third coordinate, the distance is a cumbersome task. Direct measurement of the distance is usually

not possible, but we can measure redshift which is proportional to the distance according to the Hubble law. Redshift is usually measured via spectroscopic observations by the wavelength shift of well known spectral lines. But measuring redshift is time consuming. It takes 80% of the total SDSS survey time to gather redshifts for less than 1% of the galaxies. The 5 magnitudes for all the objects is gathered in the remaining 20% of the time. We have 5 magnitudes for 270M galaxies and have redshifts for only 1M of them.

A wide range of techniques has been employed in the literature to estimate redshifts of galaxies from broadband photometric colors. Approaches have ranged from the purely empirical relations (Connolly et al. 1995a) to comparisons of the colors of galaxies with the colors predicted from galaxy spectral energy distributions (Koo 1985; Gwyn & Hartwick 1996). Although the latter, the so called template fitting technique follow better the underlying physical process, empirical approaches have their advantages, too. The color-redshift relations are derived directly from the data themselves, and are relatively free from possible systematic effects in the photometric calibration. As such, they provide a simple measure of the statistical uncertainties with the data and can demonstrate the accuracy to which we should be able to estimate redshifts once we can control the systematic errors. The simple empirical method estimates the redshift as a low order polynomial of galaxy magnitudes. Since low order polynomials cannot exactly follow the complex relation between magnitudes and redshift and higher order ones can easily overfit the data, the fitted relation is not optimal.

With the help of our spatial indexing scheme for the color space and utilizing the database server's CLR support we have developed a non-parametric photometric redshift estimation method. The idea is to estimate redshifts based on a reference set. The reference set is the catalog of 1 million galaxies where both colors and redshifts were observed by the telescope. We will refer to the other set of the about 270M objects with unknown redshifts as the unknown set. Since the reference set covers the color space relatively well, the redshift of a galaxy in the unknown set can be estimated with the redshift of the closest (with minimal Euclidean distance in color space) galaxies in the reference set. To smooth out the effects of the measurement errors we have found that instead of using the average, a local low order polynomial fit over the neighbors gives a better estimate. The pseudo code for the estimation is the following:

```
foreach (Galaxy g in UnknownSet)
{
  neighbors =
   NearestNeighbors(g, ReferenceSet)
  polynomCoeffs =
   FitPolynomial(neighbors.Colors,
                 neighbors.Redshift)
  g.Redshift = Estimate(g.colors,
   polynomCoeffs)
}
```

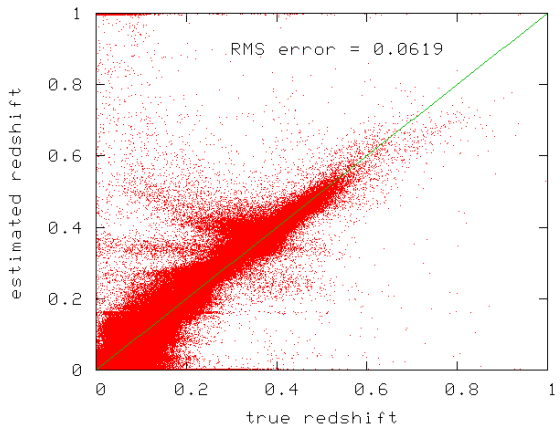

Fig. 5 (online colour at: www.an-journal.org) Redshift estimation with our original template fitting method used for the official DR5 catalog. Ideally, the estimated and true values should be equal, lying along the diagonal, but calibration problems of the templates produce large scatter.

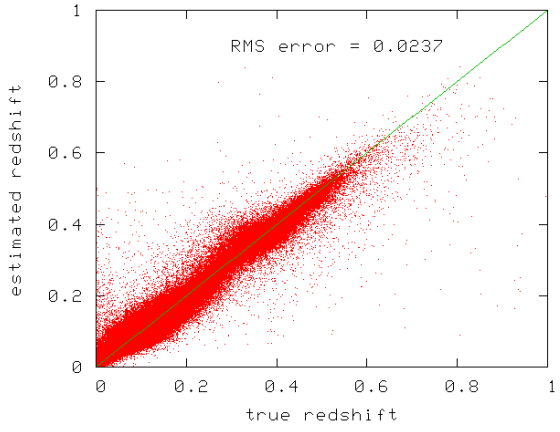

Fig. 6 (online colour at: www.an-journal.org) Redshift estimation with k-nearest neighbor polynomial fit method. Estimation error is less than half compared to the template fitting method.

In the first line of the loop, we utilize the nearest neighbor search method using our k-d tree index. The second and third line of the loop, the polynomial fit requires some intensive math calculation, including matrix inversion that would be prohibitive to do with native SQL. Also, pulling out the data from the server would impose a large time delay. Instead we have created a CLR stored procedure, a multiparameter general least square fit code written in C#, compiled and run in the SQL Server utilizing the CLR support and the AMD optimized LAPACK library. Due to the fact that the nearest neighbor fitting method is not sensitive to calibration errors, the precision of the estimation has also improved: average error decreased by more than 50%.

4.2 Searching similar spectra

To show that the utilization of the multidimensional indexing is not limited to the magnitude space, we show another

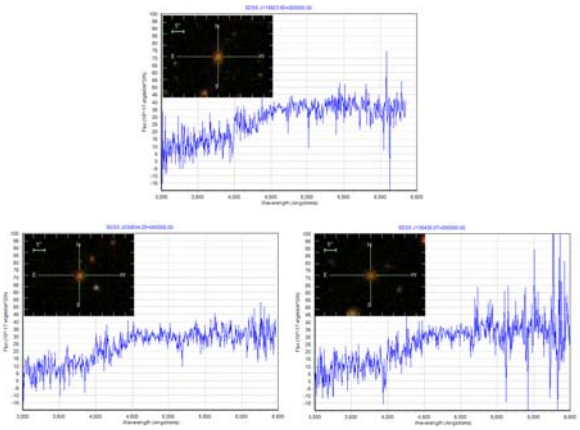

Fig. 7 (online colour at: www.an-journal.org) The top figure is a typical elliptic galaxy and its spectrum. The lower two are the two most similar spectra in the SDSS catalog based on the feature vector containing the first 5 principal components.

example that has been implemented in the Spectrum Services. Spectrum Services (http://voservices.net/spectrum) consist of easy-to-use web applications and web services for searching, plotting and managing large collections spectral energy distribution data and filter profiles as well as for performing various scientific operations on spectra. The services provide keyword search, advanced query forms and SQL interfaces for selecting spectra or band pass curves that may be retrieved in a variety of formats including XML, VOTable and ASCII. All SDSS DR1-DR5 spectra had been loaded into a database as well as other catalogs, but registered users can upload their own data making it available for the rest of the community and simulated catalogs are also available. Scientific services allow building rest-frame composite spectra; calculating synthetic magnitudes by convolving spectra with an arbitrary set of filters to generate simulated photometric catalogs on-the-fly; galactic extinction correction, fitting of the continuum and line fitting. All scientific functionality is available from the web interface and via the SOAP web services.

Spectra in SDSS are sampled at about 3000 wavelength values, so they are stored as 3000 dimensional vectors. Indexing of the 3000 dimensional space would be prohibitive, but it has been shown (Connolly et al. 1995b) that the first few principal components of the Karhunen-Loeve transform is enough to describe most of the physical characteristics. Essentially with a principal component transformation we can create a low (we have chosen 5) dimensional feature vector for galaxies. Thus, for the spectral feature space a similar index to the one we have created for the magnitude space can be built and the same stored procedures can be used for nearest neighbor searches as for the magnitude space.

Automatic searching of objects with similar spectrum can be a very useful tool for finding rare exotic objects, like high redshift quasars or brown dwarfs. We are currently working on the next version, where the most similar *simulated* spectrum can be found for an observed galaxy. Looking up the simulation parameters one can obtain the physical properties, like age, mass, metallicity and star formation history for each observed galaxy.

5 Conclusion

We have shown that the combination of multidimensional spatial indexing and modern database and web services technology provide an efficient tool for large astronomical archives. The main idea behind our web services is to move scientific functionality physically close to the database in order to spare network bandwidth. This way scientists may do research without setting up expensive hardware, downloading large data sets for days or weeks or writing and/or installing complicated software. Balázs et al.

Acknowledgements. This work was partly supported by: OTKA-T047244, MSRC-2005-038, NAP-2005/KCKHA005 and MRTN-CT-2004-503929.

References

Aurenhammer, F.: 1991, ACM Computing Surveys 23, 3, 345
Beis, J.S., Lowe, D.G.: 1997, in: *Computer Vision and Pattern Recognition*
Bentley, J.L.: 1975, Commun. ACM 18(9), 509
Chaudhary, A., Szalay, A.S., Moore, A.W.: 2002, Proc. ACM SIGMOD Workshop on Research Issues in Data Miningand Knowledge Discovery
Connolly, A.J., Csabai, I., Szalay, A.S., Koo, D.C., Kron, R.G., Munn, J.A.: 1995, AJ 110, 2655
Connolly, A.J., Szalay, A.S., Bershady, M.A., Kinney, A.L., Calzetti, D.: 1995, AJ 110, 1071
Dobos, L., Budavári, T., Csabai, I., Szalay, S.: 2004, in: F. Ochsenbein, M.G. Allen, D. Egret (eds.), *Astronomical Data Analysis Software and Systems (ADASS) XIII*, ASPC 314, p. 185
Fukugita, M., Ichikawa, T., Gunn, J.E., Doi, M., Shimasaku, K., Schneider, D.P.: 1996, AJ 111, 1748
Gray, J., Szalay, A.S., Fekete, G., et al: 2004, Microsoft Research Technical Report 32
Gwyn, S. D.J., Hartwick, F.D.A.: 1996, ApJ 468, L77
Koo, D.C.: 1985, AJ 90, 418
Kubica, J., Masiero, J., Moore, A., Jedicke, R., Connolly, A: 2005, Advances in Neural Information Processing Systems 433
Quinn, P.J., Barnes, D.G.; Csabai, I., et al.: 2004, Proceedings of the SPIE 5493, 137
Samet, H.: 2006, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann Publishers
Stoughton, C., Lupton, R., Bernardi, M., et al.: 2002, AJ 123, 485, see also http://www.sdss.org
Szalay, A.S., Gray, J.: 2006, Nature 440, 413
http://www.ivoa.net