2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

MOPED: A Scalable and Low Latency Object Recognition and Pose Estimation System

Manuel Martinez Alvaro Collet Siddhartha S. Srinivasa

***Abstract*— The latency of a perception system is crucial for a robot performing interactive tasks in dynamic human environments. We present MOPED, a fast and scalable perception system for object recognition and pose estimation. MOPED builds on POSESEQ, a state of the art object recognition algorithm, demonstrating a massive improvement in scalability and latency without sacrificing robustness. We achieve this with both algorithmic and architecture improvements, with a novel feature matching algorithm, a hybrid GPU/CPU architecture that exploits parallelism at all levels, and an optimized resource scheduler. Using the same standard hardware, we achieve up to 30x improvement on real-world scenes.**

I. INTRODUCTION

The reaction time of robots operating in dynamic environments is limited by the latency of their perception systems. Robots equipped with low latency perception systems can quickly perceive dynamic environments, enabling improved feedback control. An impressive example of such a fast reacting robot is the batting robot[1] from the Kamuro Ishikawa Laboratory which uses customized vision hardware[2] with an integrated vision chip. From a 16x16 image from the chip, the system segmented a light object against a dark background and extracted moments at a latency of 1ms.

While low latency perception systems like the aforementioned have excelled in controlled environments with relatively simple objects, real human environments that a personal service robot operates in may be cluttered, dynamic and unpredictable. These environments, like Fig. 1, for example, are characterized by their complexity and lack of structure, requiring a more general perception system.

Collet *et al.* [3] demonstrated a vision based perception system called POSESEQ capable of object recognition and full pose estimation in cluttered scenes. The system learned metric 3D models using natural (marker-free) features of objects and maintains a database. At runtime, it detected multiple objects and multiple instances of the same object from its database, and provided 6D pose estimation.

POSESEQ was designed to produce accurate pose estimates that were critical for mobile manipulation. It was robust to outliers, partial occlusions, and changes in illumination, scale and rotation. However, its latency scaled poorly with respect to the number of objects in the database and the resolution of the input image. On 640x480 images from real-world scenes, POSESEQ had a latency of about 2000ms.

Motivated by these shortcomings, we present MOPED, a system for Multiple Object Pose Estimation and Detection that improves the scalability of POSESEQ and optimizes speed (Fig. 1) without trading off its robustness and accuracy.

M. Martinez and A. Collet are with The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA - 15213, USA. {manelm, acollet}@cs.cmu.edu

Siddhartha S. Srinivasa is with Intel Labs Pittsburgh, 4720 Forbes Avenue, Suite 410, Pittsburgh, PA - 15213, USA. siddhartha.srinivasa@intel.com

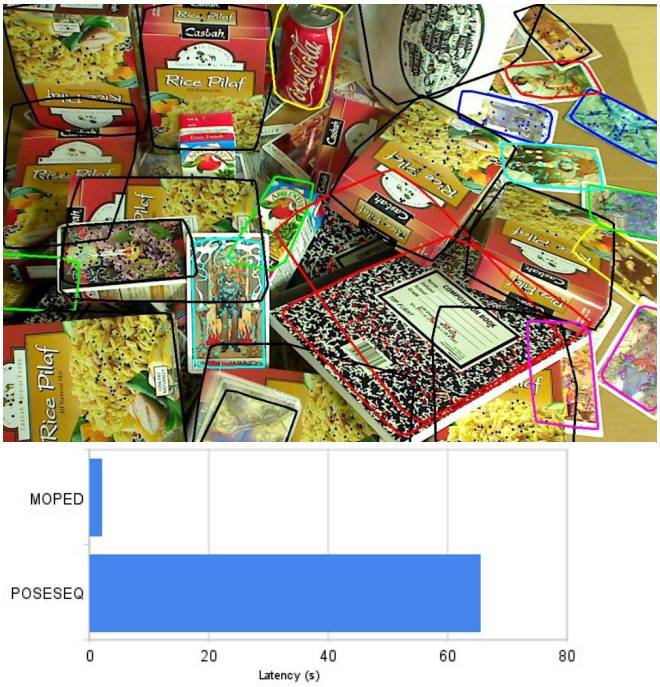

Fig. 1. A cluttered real-world scene. MOPED finds 27 objects partially-occluded, repeated and non-planar objects. The database contains 91 models and the source image is 1600x1200. In this scene, MOPED is 30.78 times faster than POSESEQ [3]

Algorithmically, MOPED uses a novel feature matching algorithm optimized for large databases with logarithmic complexity and a robust pose merging algorithm capable of efficiently rejecting outliers. Architecturally, MOPED is optimized for bandwidth and cache management and SIMD instructions. Components like feature extraction and matching have been implemented on a standard GPU. Furthermore, a novel scheduling scheme (Fig. 2) enables the efficient use of symmetric multiprocessing(SMI) architectures, utilizing all available cores on modern multi-core CPUs.

We demonstrate the speed and scalability of MOPED on a real-world object dataset of 91 objects in high clutter, as well as synthetic scenes with 400 objects. On the same 640x480 real-world images, MOPED demonstrates a latency of about 300ms., a 7x increase over POSESEQ. The gap widens to over 30x as the number of objects and the resolution of the image are scaled up.

MOPED demonstrates that novel algorithmic and architectural modifications that exploit the structure and workflow of a method can enable tremendous improvements.

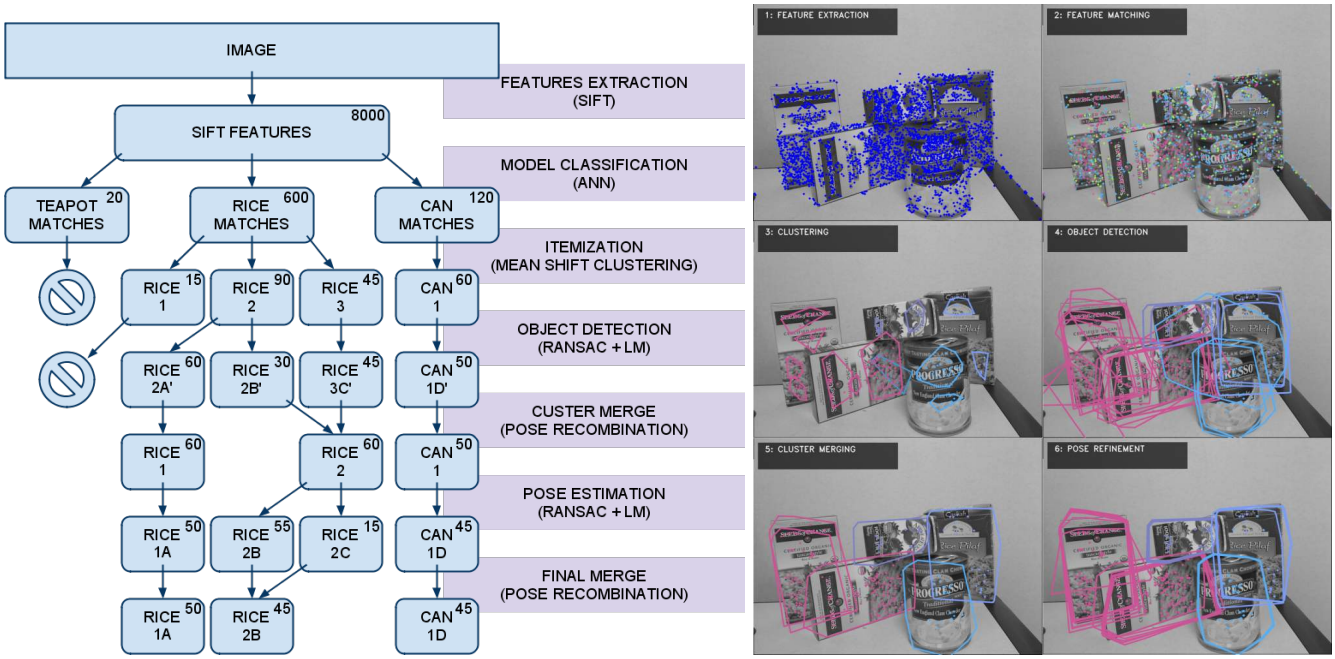

Fig. 2. The MOPED workflow and an illustration of the seven steps. Number in the upper right corner of rounded boxes represent corresponding keypoints. TEAPOT, RICE and CAN designate objects in the database. Numbers are assigned to separate instances of the same object.

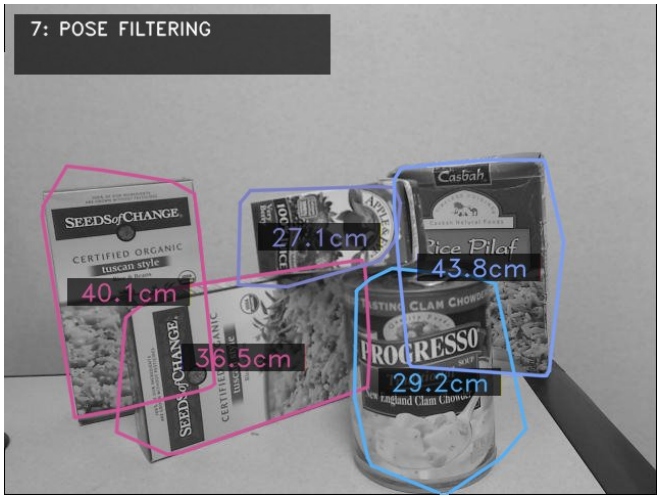

Fig. 3. Final output of MOPED after Pose Refinement

II. Related Work

The fast and efficient tracking of objects in scenes is an ongoing goal of augmented reality research, with a focus on obtaining the camera pose with respect to an object or scene, and accurately registering camera movement across frames. Gordon and Lowe [4] provide a method for accurate camera tracking using learned models of a scene and SIFT features [5]. SIFT descriptors, albeit accurate and robust, are computationally very expensive, and multiple alternative descriptors have been proposed [6–8], which in some cases allow the tracking of objects at up to 1000Hz [7]. On the other hand, object recognition and pose estimation for robotics are often explored in terms of accuracy and recognition rate [3, 9] but seldom in terms of efficiency and speed. However, recent advances in household robotics [10–13] are pushing the limits of efficiency and accuracy.

III. Algorithm Outline

MOPED starts with a precomputed database of objects learned offline. Each object consists of of 3D points with associated SIFT descriptors, constructed as described in [3]. MOPED recognizes and extracts the pose of all objects in the database (Fig. 3) that are present in the image using the following operations (also detailed in Fig. 2):

1. Feature Extraction. Extract salient features from image.

2. Feature Matching. Create correspondences between extracted features in the image and object features stored in the database. For efficiency, approximate matching techniques are used, but produce more outliers.

3. Keypoint Clustering. Cluster in image space features matched to a particular object. Spatially close features are more likely to belong to the same object instance.

4. Coarse object detection. Process each cluster independently in search of objects. RANSAC and Levenberg-Marquardt (LM) are used to find object instances that are loosely consistent with each object's geometry in spite of outliers. The number of RANSAC and LM iterations are kept low to accept very coarse object detections.

5. Cluster merging. As the same object might be present in multiple clusters, re-cluster image space features using poses resulting from Step 4. New, larger clusters are created, that often contain all consistent features for a whole object.

6. Fine object detection. After Steps 4 and 5, most outliers have been removed, and it is reasonable to assume that each of the new clusters contain features corresponding to only one instance of an object. Repeat LM and RANSAC with a larger number of iterations to estimate a single pose from each cluster.

7. Pose Filtering. A final merging step removes any multiple detection that might have survived, by again merging together object instances with similar poses.

Fig. 4. MOPED Benchmarks. For the sake of clarity, only half of the detected objects are marked. (a) The Rotation Benchmark: MOPED processes this scene 36.4x faster than POSESEQ. (b) The Zoom Benchmark: MOPED processes this scene 23.4x faster than POSESEQ. (c) The Simple Movie Benchmark. (d) The Complex Movie Benchmark.

IV. BENCHMARKS

We present four novel benchmarks (Fig. 4) designed to stress test every component of our system. We performed all experiments on a 2.33GHz quad-core Intel(R) Xeon(R) E5345 CPU, 4 GB of RAM and a nVidia GeForce GTX 260 GPU running Ubuntu 8.04 (32 bits).

A. The Rotation Benchmark

The Rotation Benchmark is a set of synthetic images that contains highly cluttered scenes with up to 400 cards in different sizes and orientations. This benchmark is designed to test MOPED's scalability with respect to the database size, while keeping a constant number of features and objects. We have generated a total of 100 independent images for different resolutions (1400×1050, 1000×750, 700×525, 500×375 and 350×262). Each image contains from 5 to 80 different objects and up to 400 simultaneous object instances.

B. The Zoom Benchmark

The Zoom Benchmark is a set of synthetic images that progressively zooms in on 160 cards until only 12 cards are visible. This benchmark is designed to check the scalability of MOPED with respect to the total number of detected objects in a scene. We generated a total of 145 independent images for different resolutions (1400×1050, 1000×750, 700×525, 500×375 and 350×262). Each image contains from 12 to 80 different objects and up to 160 simultaneous object instances. This benchmark simulates a board with 160 cards seen by a 60° FOV camera at distances ranging from 1100mm to 300mm. The objects were chosen to have the same number of features at each scale. Each image has over 25000 features.

C. The Simple Movie Benchmark

Synthetic benchmarks are useful to test a system in controlled conditions, but are a poor estimator of the performance of a system in the real world. Therefore, we provide two real-world scenarios for algorithm comparison. The Simple Movie Benchmark consists of a 1900-frame movie at 1280 x 720 resolution, each image containing up to 18 simultaneous object instances.

D. The Complex Movie Benchmark

The Complex Movie Benchmark consists of a 3542-frame movie at 1600 x 1200 resolution, each image containing up to 60 simultaneous object instances. The database contains 91 models and 47342 SIFT features when running this benchmark. It is noteworthy that the scenes in this video present particularly complex situations, including: several objects of the same model contiguous to each other, which stresses the clustering step; overlapping partially-occluded objects, which stresses RANSAC; and objects in particularly ambiguous poses, which stresses both LM and the merging algorithm, that encounter difficulties determining which pose is preferable.

TABLE I
SIFT VS. SURF: RECOGNITION PERFORMANCE IN ZOOM BENCHMARK.

	Avg. Processing Time (ms)	Avg. Recognized Objects
SIFT	223.072	13.83
SURF	86.136	6.27

V. ALGORITHMIC IMPROVEMENTS

A. Feature Extraction

The most computationally expensive step of MOPED is the extraction of point features from each new image, for which the original POSESEQ used a CPU-optimized version of SIFT. We considered SURF features[6], considered as a fast alternative to SIFT. Table I compares the usage of SURF vs. SIFT in terms of computation time and object recognition performance. SURF proves to be 2.59x faster than SIFT at the cost of detecting 54% less objects. In addition, the performance gap between both methods decreases significantly as the size of the images increases, as shown in Fig. 7. On our benchmarks we found SIFT to be the almost always the better alternative.

B. Feature Matching

Computing the correspondences between image features and the object database can be expensive. Matching is done in the 128-dimensional space of SIFT features. Depending on the number of objects, the database can contain over 50,000 features. Depending on the resolution and complexity of the scene, the image can contain over 10,000 features. Approximate approaches to compute correspondences build *kd-trees* out of sets of points. POSESEQ, following [5], uses Approximate 2-Nearest Neighbors (2-ANN) and performs a distance ratio test between the first 2 NNs to remove outliers. A kd-tree is built for each model in the database once offline, and is independently matched against every new image, with a complexity of $\mathcal{O}(F_{im} M_{db} \log(F_m))$, where F_{im} is the number of features on the image, M_{db} the number of models in the database, and F_m the mean number of features for each model.

When M_{db} is large, this approach is vastly inefficient as the cost of accessing each object kd-tree dominates. A naïve alternative, which we term *SIMPLE*, builds just one *kd-tree* containing the features from all models. This solution has a complexity of $\mathcal{O}(F_{im} \log(M_{db} F_m))$. However, the distance ratio is not an adequate measure when using such a large number of features, because of the presence of similar features in different objects.

Alternatively, one can consider a k-ANN approach (with $k > 2$). k-ANN implementations using kd-trees can provide more neighbors without significantly increasing their computational cost, as they are often a byproduct of the process of obtaining the nearest neighbor. The distance ratio is then applied to the 2 nearest neighbors from the same model, if available. If the nearest neighbor is the only neighbor for a given model, we apply the distance ratio with the next neighbor on the list. This algorithm is the default choice for MOPED.

Finally, MOPED also supports a GPU-based exact feature matching algorithm. The parallel nature of the brute force matching algorithm suits the GPU, and allows it to be faster than the ANN approach when F_m is not too large. Given that this algorithm scales linearly with the number of features instead of logarithmically, we can match each model independently without performance loss.

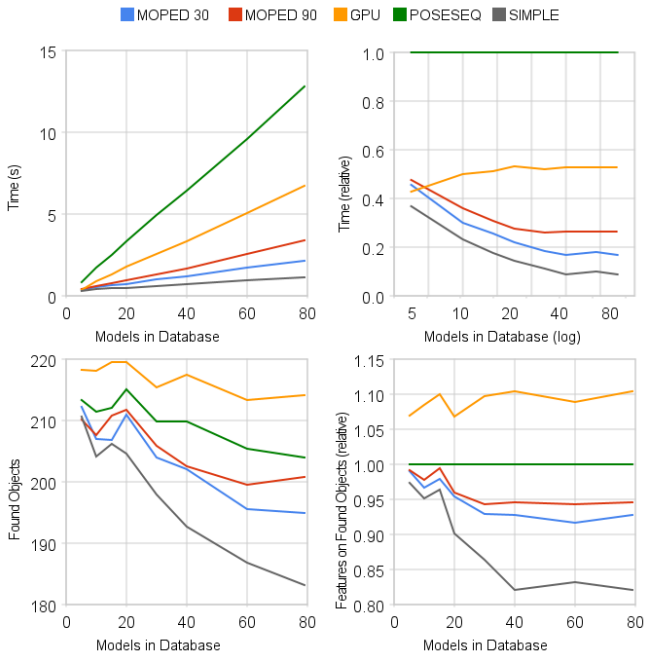

Fig. 5. Scalability of feature matching algorithms with respect to the size of the database, in the Rotation Benchmark 1400 × 1050 resolution).

TABLE II
FEATURE MATCHING ALGORITHMS IN THE SIMPLE MOVIE BENCHMARK.

Correspondences:	After Matching	After clustering	Final
GPU	3893.7	853.2	562.1
POSESEQ	**3893.6**	**712.0**	**449.2**
SIMPLE	1778.4	508.8	394.7
MOPED	3624.9	713.6	428.9

	Matching Time(ms)	Objects Found
GPU	253.34	8.8
POSESEQ	498.586	8.0
SIMPLE	**129.85**	7.5
MOPED	140.36	**8.2**

Fig. 5 compares the cost of the different alternatives on the Rotation Benchmark. POSESEQ and GPU scale linearly with respect to M_{db}, while SIMPLE and MOPED scale almost logarithmically. We show MOPED using $k = 90$ and $k = 30$. The value of k adjusts the speed and quality behavior of MOPED between POSESEQ ($k = \infty$) and SIMPLE ($k = 2$). The recognition performance of MOPED when using the different strategies is shown in Table II. GPU provides the ground truth as it is exhaustive. POSESEQ comes closest in raw matching accuracy with MOPED a close second. However, the number of objects detected are nearly the same. The matching speed of MOPED is, however, significantly better than POSESEQ. Feature matching in MOPED thus provides a big speed increase without sacrificing much accuracy.

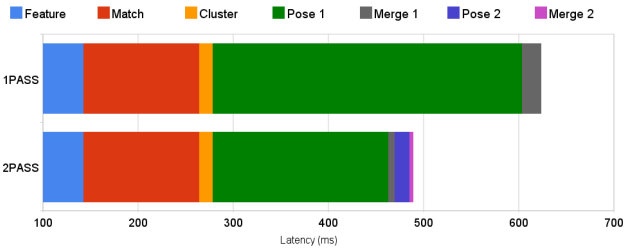

Fig. 6. Total latency using single pass vs. double pass pose estimation, in the Simple Movie Benchmark.

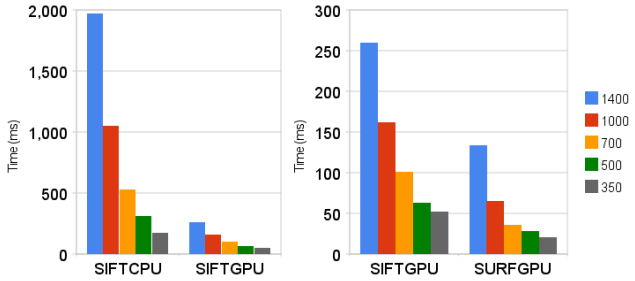

Fig. 7. SIFT-CPU vs. SIFT-GPU vs. SURF-GPU, in the Rotation Benchmark at different resolutions. (left) SIFT-CPU vs. SIFT-GPU: 658% performance increase in SIFT extraction on GPU. (right) SIFT-GPU vs. SURF-GPU: SURF is 91% faster than SIFT at the cost of lower matching performance.

C. Decoupling Detection and Pose Estimation

The duplicated RANSAC-LM steps implemented in MOPED represent an important advantage over the single-step detection of POSESEQ. The first pass (Coarse Object Detection) uses a low number of LM iterations, detecting hypotheses with a coarse pose. The second pass (Fine Object Detection) is performed only after filtering most outliers and merging clusters together, so we use a higher number of iterations to estimate object poses with high precision. The combined result when tested on all benchmarks produced equivalent robustness and precision, but required fewer total iterations. A representative test on the Simple Movie Benchmark is shown in Fig. 6.

VI. Architecture Optimizations

Our algorithmic improvements were focused mainly on boosting the scalability and robustness of the system. The architectural improvements of MOPED are obtained as a result of a re-implementation designed to make the best use of all the processing resources of standard compute hardware. In particular, we use GPU-based processing, intra-core parallelization using SIMD instructions, and multi-core parallelization in coarse grained algorithms. The memory subsystem, including bandwidth transfer and cache managing, has also been carefully optimized.

All optimizations have been devised to reduce the latency between the acquisition of an image and the output of the pose estimates, to enable faster response times from our robotic platform.

A. GPU and Embarrassingly Parallel Problems

State-of-the-art CPUs have a peak performance of 12.8 GFLOPS, which can be extended to 76.8 GFLOPS if using vectorization instructions like SSE and Single Precision (SP) Floating Point. State-of-the-art GPUs have a theoretical maximum performance of more than 2000 SP GFLOPS.

To use GPU resources efficiently, input data needs to be transferred to the GPU memory. Then, algorithms are executed simultaneously on all *shaders*, and finally recover the results from the GPU memory. As communication between shaders is expensive, the best GPU-performing algorithms are those that can be divided evenly into a large number of simple tasks. This class of easily separable problems is called *Embarrassingly Parallel Problems (EPP)*.

1) SIFT vs. SURF on GPU: Most feature extraction algorithms consist of an initial keypoint detection step followed by a descriptor calculation for each keypoint, and both of them are *EPP*. Keypoint searching algorithms can process each pixel from the image independently. They may need information about neighboring pixels, but they do not need *results* from them. After obtaining the list of keypoints, the respective descriptors are also calculated independently.

MOPED uses *SIFTGPU*[8] as its main feature extraction algorithm. MOPED supports *GPU-SURF*[14], but it is not used by default as it is less robust than SIFT. If compatible graphics hardware is not detected MOPED automatically reverts back to performing SIFT extraction on the CPU, which is an OpenMP-enabled, CPU-optimized version of SIFT. We compare the latency of the three implementations in Fig. 7. The comparison is as expected: GPU versions of both SIFT and SURF provide tremendous improvements over their non-GPU versions. We were particularly impressed with the almost tenfold increase in speed with SIFTGPU.

2) GPU Matching: Performing feature matching in the GPU requires a different approach than the standard Approximate Nearest Neighbor techniques. Using ANN, each match involves searching in a kd-tree, which requires fast local storage and a heavy use of branching that are not suitable for GPUs.

Instead of using ANN, [8] suggest the use of brute force nearest neighbor search on the GPU, which scales quite well as vector processing matches perfectly the GPU structure. In Fig. 5, brute force GPU matching is shown to be faster than ANN and provide better quality matches because it is not approximate. We believe that as graphics hardware becomes cheaper and more powerful, brute-force feature matching might be the inevitable choice.

B. Intra-core optimizations

SSE instructions allow MOPED to perform 12 floating point instructions per cycle instead of just one. The 3D to 2D projection function, critical in the pose estimation steps, is massively improved by using SSE-specific algorithms from[15][16].

The memory footprint of MOPED is very lightweight for current computers. In the case of a database of 100 models and a total of 102.400 SIFT features, the required memory is less than 13MB. Runtime memory footprint is also small: a scene with 100 different objects with 100 matched features each would require less than 10 MB of memory to be processed. This is possible thanks to using dynamic and compact structures, such as lists and sets, and removing unused data as soon as possible. In addition, SIFT descriptors are stored as integer numbers in a 128-byte array

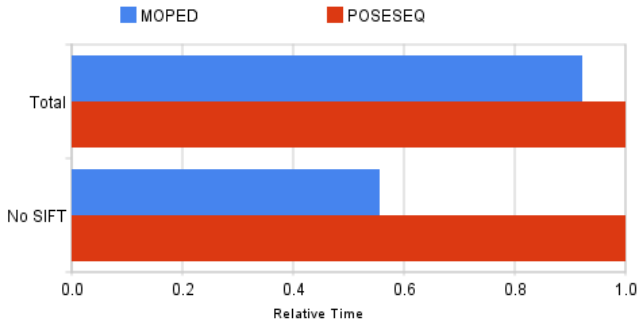

Fig. 8. Intra-CPU performance of MOPED relative to POSESEQ, in the Complex Movie Benchmark. (top) Total time/frame relative to POSESEQ. (bottom) Time/frame without counting SIFT extraction.

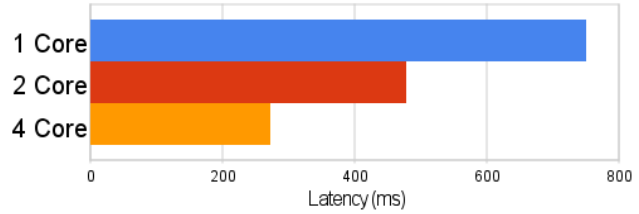

Fig. 9. Pose estimation performance in multi-core CPUs, in the Complex Movie Benchmark.

instead of a 512-byte array. Cache performance has been greatly improved due to the heavy use of memory-aligned and compact data structures [17].

The main data structures are kept constant throughout the algorithm, so that no data needs to be copied or translated between steps. k-ANN feature matching benefits from compact structures in the kd-tree storage, as smaller structures increase the probability of staying in the cache for faster processing. In feature clustering, the performance of Mean Shift is boosted 250 times through the use of compact data structures.

The overall performance increase is over 67% in CPU processing tasks (see Fig. 8).

C. Symmetric Multiprocessing

Symmetric Multiprocessing (SMP) is a multiprocessor computer architecture with identical processors and shared memory space. Most multi-core based computers are SMP systems. *OpenMP* is a framework to use multi-processing in SMP systems that we implement in MOPED.

We use *Best Fit Decreasing* to balance the load between the cores using the size of a cluster as an estimate of its processing time, given that each cluster of features can be processed independently. Tests on a subset of 10 images from the Complex Movie Benchmark show performance improvements of 55% and 174% on dual and quad core CPUs respectively (see Fig. 9).

D. Multi Frame Scheduling

In order to maximize the system throughput, MOPED can benefit from GPU-CPU pipeline scheduling[18]. In order to use all available computing resources, a second execution thread can be added, as shown in Fig. 10. However, the GPU and CPU execution times are not equal in real scenes, and one of the execution threads often needs to wait for the

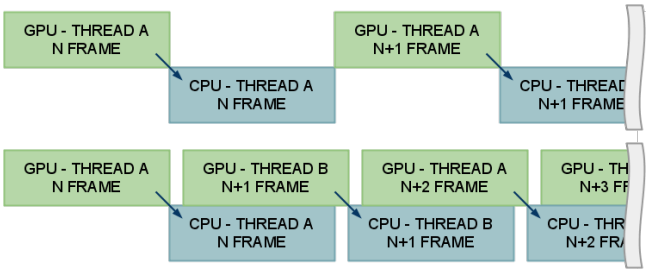

Fig. 10. (top) Standard MOPED uses the GPU to obtain the SIFT features, and then uses the CPU to process them. (bottom) Addition of a second execution thread does not substantially increase the system latency.

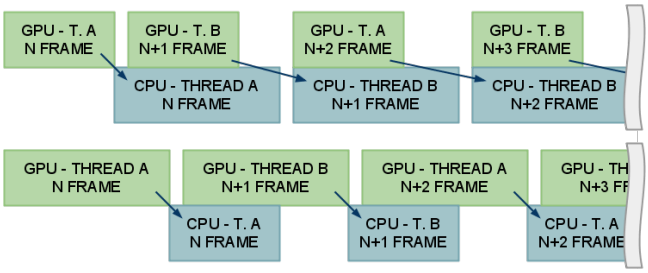

Fig. 11. (top) Limiting factor: CPU. GPU thread processing frame N+1 must wait for CPU processing frame N to finish, increasing latency. (bottom) Limiting factor: GPU. No substantial increase in latency.

other (see Fig. 11), so the latency can increase significantly, especially if using high resolution images (see Fig. 12).

VII. Scalability

The Simple Movie Benchmark consists of a 1900-frame movie at 1280 x 720 resolution, each image containing up to 18 simultaneous object instances. This is the type of scene that our robot HERB [10] encounters daily in its environment. Using a database of 11 models, our results show a 5.74x speed increase using Standard MOPED and a 7.44x speed increase with Pipelined MOPED (see Table III). Mean latency is 303ms and 368ms, respectively.

The Complex Movie Benchmark consists of a 3542-frame movie at 1600 x 1200 resolution, each image containing up to 60 simultaneous object instances. The Complex Movie Benchmark contains extreme clutter, seldom seen in the real world. MOPED shines in this scenario, outperforming POSESEQ by over 30x (Table III). This demonstrates the extreme scalability of MOPED. This is further reinforced in the synthetic benchmarks. In Fig. 13 and Fig. 14, we

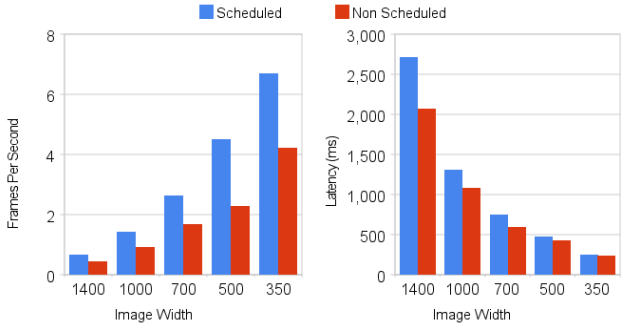

Fig. 12. Impact of Pipeline Scheduling. (left) Throughput (FPS) comparison. (right) Added latency using pipeline scheduling. Since GPU processing is the bottleneck on very small resolutions, these are the best scenarios for pipeline scheduling. At 500x380, throughput is increased by 95.6% and latency is increased by 9%.

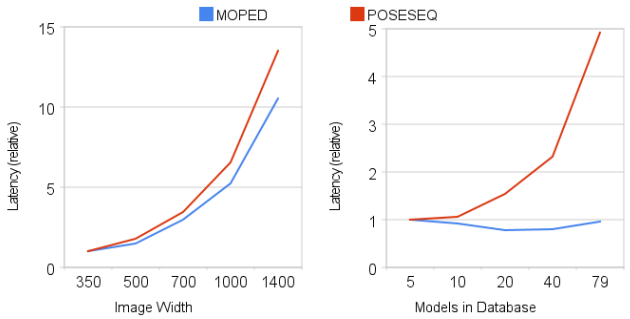

Fig. 13. Scalability experiments in the Rotation Benchmark. (left) Latency with respect to image resolution. (right) Latency with respect to database size.

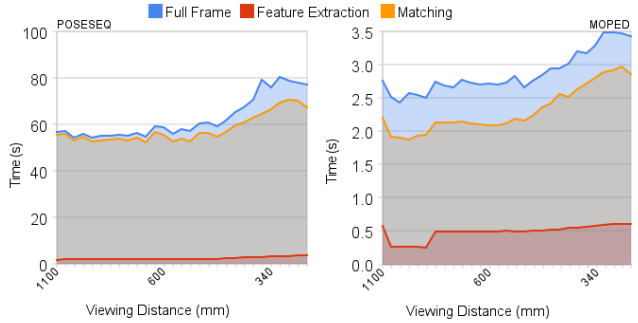

Fig. 14. Scalability with respect to the number of objects in the scene in the Zoom Benchmark. There are 160 small objects at 1100mm and only 12 large objects at 300mm. (left) Scalability of POSESEQ. (right) Scalability of MOPED.

show the processing time for SIFT extraction and model matching as well as the total processing time per frame for both methods. MOPED's flat latency curve is particularly encouraging.

Our experiments show that both MOPED and POSESEQ scale quadratically in execution time with respect to image resolution. However, POSESEQ's performance is highly dependent on the number of different objects in the database, while MOPED's performance is almost constant (Fig. 13).

The Zoom Benchmark shows a relatively constant number of detected features (Fig. 14), although 160 objects are detected at 1100mm and only 12 are visible at 300mm. It is interesting to notice that the required time is inversely proportional to the number of objects in the image, i.e. a small number of large objects are more demanding than large numbers of small objects. In addition, SIFTGPU exhibit bimodal behavior at the memory limit of the graphics card.

TABLE III
PERFORMANCE IN SIMPLE AND COMPLEX MOVIE BENCHMARKS.

Simple Movie	FPS	Latency (ms)	Latency Sd (ms)
Pipelined MOPED	3.49	368.445	92.3431
Standard MOPED	2.70	303.229	69.2581
POSESEQ	0.47	2124.30	286.538

Complex Movie	Latency (ms)
MOPED	2173.83
POSESEQ	65568.2

VIII. CONCLUSIONS

We have demonstrated MOPED, an algorithmic and architectural evolution of the state-of-the-art POSESEQ object recognition and pose estimation algorithm. MOPED is scalable, fast, and robust, utilizing all of the processing power, in- and out-of-core, available in modern computers. As a result, it achieves low latency and high scalability, enabling robots to perceive and interact in cluttered dynamic scenes.

IX. ACKNOWLEDGMENTS

This material is based upon work partially supported by the National Science Foundation under Grant No. EEC-0540865. Alvaro Collet is partially supported by Caja Madrid fellowship. Special thanks to the members of the Personal Robotics project at Intel Labs Pittsburgh, Lily Mummert and Babu Pillai for useful discussions and comments.

REFERENCES

[1] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator." in *IEEE ICRA*, 2006.
[2] A. Namiki, T. Komuro, and M. Ishikawa, "High-speed sensory-motor fusion for robotic grasping," *Measurement Science and Technology*, vol. 13, pp. 1767–1778, Nov. 2002.
[3] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object recognition and full pose registration from a single image for robotic manipulation," in *IEEE ICRA*. Kobe: IEEE, May 2009, pp. 48–55.
[4] I. Gordon and D. G. Lowe, "What and where: 3d object recognition with accurate pose," in *Toward Category-Level Object Recognition*, 2006, pp. 67–82.
[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, pp. 91–110, 2004.
[6] H. Bay, T. Tuytelaars, and A. L. Van Gool, "Surf: Speeded up robust features," in *ECCV*, 2006.
[7] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose tracking from natural features on mobile phones," in *Mixed and Augmented Reality*, 2008.
[8] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," http://cs.unc.edu/ ccwu/siftgpu, 2007.
[9] F. Vikstén, R. Söderberg, K. Nordberg, and C. Perwass, "Increasing Pose Estimation Performance using Multi-cue Integration," in *IEEE ICRA*, 2006.
[10] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. C. Romea, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and J. M. Vandeweghe, "Herb: a home exploring robotic butler," *Auton. Robots*, vol. 28, no. 1, pp. 5–20, Jan. 2010.
[11] A. Saxena, J. Driemeyer, and A. Ng, "Robotic Grasping of Novel Objects using Vision," *IJRR*, vol. 27, no. 2, pp. 157–173, 2008.
[12] "The pr platform," http://pr.willowgarage.com, 2008.
[13] H. Nguyen, C. Anderson, A. Trevor, A. Jain, Z. Xu, and C. Kemp, "El-e: An Assitive Robot that Fetches Objects from Flat Surfaces," in *Proc. Human Robot Interaction*, 2008.
[14] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *IEEE CVPR*, 2008.
[15] J. van Waveren, "From quaternion to matrix and back," 2005.
[16] G. Conte, S. Tommesani, and F. Zanichelli, "The long and winding road to high-performance image processing with MMX/SSE," in *Proceedings of the Fifth IEEE Int. Wshp. on Comp. Architectures for Machine Perception*, 2000, p. 302.
[17] T. J. Dysart, B. J. Moore, L. Schaelicke, and P. M. Kogge, "Cache implications of aggressively pipelined high performance microprocessors," in *Int. Sym. on Performance Analysis of Systems and Software*, 2004.
[18] K. S. Chatha and R. Vemuri, "Hardware-software partitioning and pipelined scheduling of transformative applications," *IEEE Trans. VLSI*, 2002.