AI learns to play MARIO

Vijay Dev Reddy Chevireddi
University of Maryland

Venkata Sai Charan Paladugu
University of Maryland

***Abstract*—In this project, we explore the application of advanced reinforcement learning techniques to teach an artificial intelligence (AI) model to autonomously navigate and play the game "Super Mario Bros". We employed a combination of Double Deep Q-Network (DDQN) and state-of-the-art neural network architectures—specifically, ResNet-18 alongside the SWIN Transformer—to enhance the model's learning capabilities and improve its decision-making process in complex, dynamic environments. The environment for this experiment was set up using the OpenAI Gym's "SuperMarioBros-v0", where we limited the action space to five discrete actions to optimize performance. The observation input to the model was a processed 4x64x64 pixel representation, capturing temporal context over sequential frames. This setup facilitated the model's ability to generalize from visual data to effective in-game actions. Our model demonstrates significant improvement in gameplay strategy and efficiency compared to conventional CNN-based approaches, highlighted by superior navigation skills and higher cumulative rewards. This project not only furthers the understanding of applying hybrid deep learning models in video game environments but also enhances the broader capabilities of AI in complex decision-making scenarios.**

I. Introduction

The challenge of developing artificial intelligence (AI) that can interpret and interact with complex environments has been a focal point of research within the machine learning community. Video games, particularly those with rich, dynamic worlds like *Super Mario Bros*, provide a fertile ground for exploring and advancing AI technologies. These games demand strategic thinking, quick decision-making, and adaptability—traits that are crucial for AI applications beyond gaming, such as autonomous driving and advanced robotics.

Super Mario Bros is not just a popular video game; it represents a microcosm of real-world problems AI might face, such as dynamic obstacle navigation, pattern recognition, and predictive planning. This project taps into this potential by employing a sophisticated AI model using a blend of Double Deep Q-Network (DDQN) and innovative neural network architectures, namely ResNet-18 and SWIN Transformer. The DDQN helps mitigate issues associated with the stability of Q-values in reinforcement learning, while the combined strength of ResNet-18 and SWIN Transformer provides a robust framework for processing and interpreting complex visual inputs.

The rationale behind integrating these advanced technologies lies in their proven capabilities in their respective fields. ResNet-18 has established its efficiency in handling deep learning tasks involving image recognition due to its deep residual learning framework. On the other hand, the SWIN Transformer introduces an attention mechanism that optimally allocates computational resources to different parts of the input image, a critical feature for enhancing the decision-making process in games where situational awareness from visual data is key.

A. Objectives

The project is structured around several key objectives:

1) **To implement and refine a deep reinforcement learning model** capable of mastering *Super Mario Bros* through purely end-to-end learning from pixel data.
2) **To investigate the synergy between DDQN, ResNet-18, and SWIN Transformer** in handling the complexities of a multi-faceted game environment, aiming to set a new benchmark in AI gameplay.
3) **To evaluate the strategic depth and learning curve** of the AI by analyzing its gameplay, identifying patterns and decision-making strategies that emerge as it learns.
4) **To draw parallels between AI behaviors in game settings and potential real-world applications**, offering insights into how similar models could be adapted for use in areas such as autonomous navigation and real-time decision-making systems.

B. Significance

This project's significance is twofold. Primarily, it contributes to the academic and practical understanding of applying hybrid AI models to complex, dynamic tasks. Secondly, it serves as a proof of concept that the methodologies developed for *Super Mario Bros* can inform broader AI strategies applicable in real-world scenarios, pushing the boundaries of what AI can achieve in environments that mimic the unpredictability and complexity of human settings.

By meeting these objectives, this study not only enhances the knowledge base of AI-driven gameplay but also illuminates the path toward more generalized AI applications, marking a significant step forward in the journey towards creating truly adaptive and intelligent systems.

II. Literature Review

The exploration of reinforcement learning (RL) in video games serves as a rich field for demonstrating and enhancing the capabilities of artificial intelligence systems. A foundational advancement in this area was achieved by Mnih et al. (2013) in their groundbreaking paper, "Playing Atari

with Deep Reinforcement Learning." This work introduced the Deep Q-Network (DQN), which leveraged convolutional neural networks to learn optimal policies directly from high-dimensional sensory inputs. Their method demonstrated, for the first time, that a deep learning model could successfully learn control policies from raw video data in complex game environments, notably those provided by the Atari 2600 games. This seminal research underscored the potential of deep learning models to master complex video games through end-to-end learning from pixels alone, setting the stage for subsequent research into sophisticated RL algorithms.

A. Previous Research and Theoretical Foundations

Following the introduction of DQN, the field saw developments in algorithms designed to enhance the stability and efficiency of RL in video gaming contexts. Variants of Q-learning, such as Double Q-learning (van Hasselt et al., 2010), aimed to address issues like the overestimation of Q-values. Further enhancements led to the Double Deep Q-Network (DDQN), introduced by van Hasselt et al. (2016), which improved upon the original DQN by decoupling the selection and evaluation of the action-value function, leading to more stable and reliable training processes.

B. Consideration of Alternative Algorithms

Alternative algorithms were considered for this project:

- **Proximal Policy Optimization (PPO)**: Known for its robust training behavior, as described by Schulman et al. (2017), PPO is computationally expensive for complex environments like Super Mario Bros.
- **Asynchronous Actor-Critic Agents (A3C)**: Although efficient in learning multiple policy and value functions, A3C, proposed by Mnih et al. (2016), can be difficult to tune across different game scenarios.
- **Monte Carlo Tree Search (MCTS)**: While effective in strategy games like Go (Silver et al., 2017), MCTS is less applicable for real-time decision-making from raw pixels.

C. Rationale for Algorithm Choice

DDQN was chosen due to its ability to efficiently manage high-dimensional input spaces and balance performance with computational efficiency, crucial for the dynamic environments in Super Mario Bros. This choice was also informed by the foundational work of Mnih et al. (2013), which demonstrated the applicability of deep Q-learning to video games.

D. Gaps in Existing Literature

Despite extensive research on RL in gaming, combining these methods with advanced neural architectures like SWIN Transformers is underexplored, particularly in video gaming contexts where the focus is typically on conventional CNNs.

E. Contribution to Knowledge

This project seeks to fill the existing gap by integrating DDQN with a hybrid neural network architecture combining ResNet-18 and SWIN Transformer. This approach is anticipated to provide superior performance over standard CNN-based models, offering new insights into both the strategic gameplay elements and the broader application of transformer models in real-time decision-making scenarios.

III. Methodology

This section presents the comprehensive methods employed to develop and validate an AI agent capable of playing "Super Mario Bros" using a Double Deep Q-Network (DDQN) integrated with advanced neural architectures. The methodology includes environment setup, data collection, neural network configurations, training procedures, and evaluation metrics.

A. Environment Setup and Data Collection

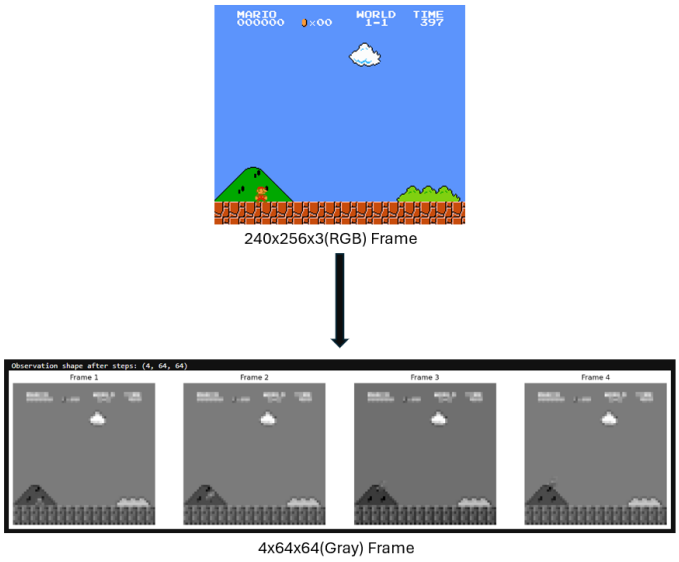

Fig. 1. Frames Wrapping

The game environment is customized through several wrappers to optimize the learning process and manage computational resources:

1) **Max Skip Environment:** Reduces the frequency of decision points by executing the same action across multiple frames and accumulating rewards, which speeds up the game's progress without sacrificing learning opportunities.
2) **Frame Processing:** Transforms game frames to a consistent 64x64 grayscale format, simplifying the input while preserving essential visual information.
3) **PyTorch Image and Buffer Wrappers:** Adjust the input for compatibility with PyTorch and maintain a buffer of the last four frames to capture temporal dependencies, which are critical for understanding motion and changes over time.

B. Action Space

In our project, we substantially reduced the action space from the original "SuperMarioBros-v0" environment, which offered 256 discrete actions, to just five key actions. This reduction was aimed at enhancing the learning efficiency and performance of our AI model.

Motivation for Reducing the Action Space: The extensive action space provided by the OpenAI Gym environment allows for intricate control and complex gameplay strategies. However, this complexity can significantly hinder the AI's training process due to the increased computational demand and the extended time required to explore and learn optimal actions. Our goal was to simplify the AI's decision-making process to accelerate training and improve efficiency.

Methodology for Action Space Reduction: We scrutinized the game mechanics and identified the most impactful actions necessary for effective gameplay. The actions retained were strategically chosen to encompass all necessary movements for navigating through the game's levels effectively without overcomplicating the decision process. The selected actions included:

- **Right:** Moves Mario to the right, the primary movement direction.
- **Jump Right:** Combines jumping and moving right, crucial for overcoming obstacles.
- **Sprint Right:** Enables faster movement to the right, useful for quick escapes and long jumps.
- **Jump:** A vertical jump, essential for reaching higher platforms and interacting with overhead blocks.
- **No Operation (No-op):** Represents inaction, strategically important for timing and pauses.

Impact of Action Space Reduction: The reduction to these five actions significantly streamlined the training process, yielding several benefits:

- **Faster Convergence:** The AI model learned optimal strategies more quickly due to fewer action choices at each decision point.
- **Improved Performance:** The model made more strategic and effective decisions, enhancing gameplay success.
- **Resource Efficiency:** The training became less demanding in terms of computational resources and memory, facilitating a more efficient learning cycle.

This strategic reduction of the action space not only simplified the training but also empowered the AI to achieve higher performance levels, demonstrating the effectiveness of focused learning strategies in complex environments.

C. Neural Network Architecture

DDQN Agent:

- The DDQN architecture is built upon two main networks—the predicted Q-network and the target Q-network—which help stabilize learning by decoupling the learning of Q-values.
- **Network Details:**
  - **Input:** Receives a state space from the environment.
  - **Output:** Provides Q-values for each possible action, which inform the agent's decisions.

DQNSolver:

- This neural network model serves as the backbone for both the predicted and target Q-networks.
- **ResNet-18 Base:** Modified to accommodate the unique requirements of the game's visual input. It initially processes images to extract features using convolutional layers.
- **SWIN Transformer:** Integrated after initial feature extraction by ResNet-18, this module enhances the model's ability to focus on and interpret relevant parts of the input image. The SWIN Transformer employs a self-attention mechanism, which allows the model to weigh parts of the input differently, improving its ability to recognize important features and patterns that are crucial for strategic decision-making in complex environments.

1) SWIN Transformer and the Shifted Window Concept: The SWIN Transformer is a specialized type of vision transformer designed for optimal performance in image-related tasks. Its unique feature, the shifted window mechanism, enhances the model's ability to handle complex visual inputs efficiently. Here's an in-depth look at this concept and its benefits.

a) Basic Window Partitioning: In conventional transformers, self-attention involves computing relationships across all pairs of input positions, which can be computationally prohibitive for large images. The SWIN Transformer circumvents this by dividing the image into non-overlapping local windows wherein self-attention is computed. This initial partitioning reduces complexity by focusing on local interactions.

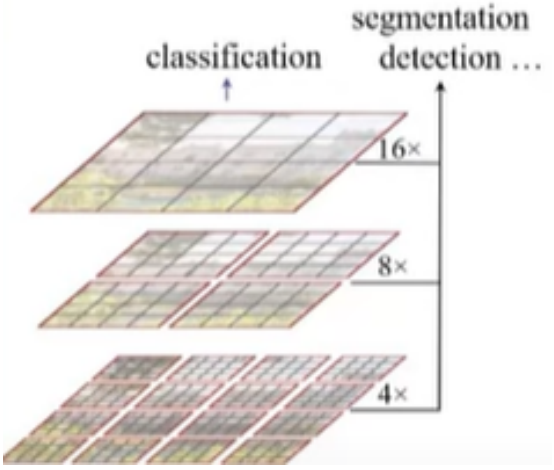

Fig. 2. Basic Window Partitioning

b) Shifted Window Mechanism: The shifted window mechanism is pivotal in allowing the SWIN Transformer to model both local and more extended spatial relationships without incurring a high computational cost:

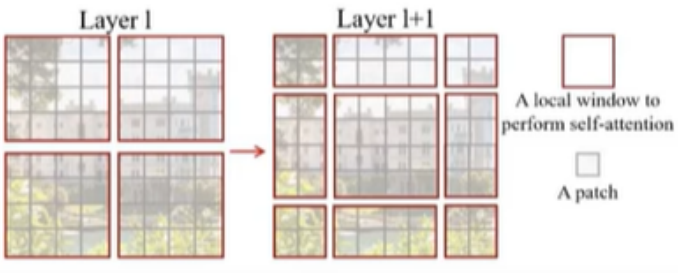

Fig. 3. Shifted Window Mechanism

1) **Initial Layer Partitioning:** The image is first partitioned into fixed-size windows, and self-attention is computed within each window independently.
2) **Window Shifting:** In subsequent layers, these windows are shifted by a set number of pixels, typically half the window size. This shift alters the grouping of pixels and their local neighborhoods.
3) **Alternating Layers:** By alternating between the original and shifted window configurations across layers, the model ensures comprehensive coverage of interactions across the entire image.

c) Benefits of Shifted Windows: The shifted window design offers several advantages:

- **Reduced Computational Load:** It localizes the computationally intensive self-attention process to smaller subsets of the image, thereby reducing the overall computational demands.
- **Enhanced Representation Capabilities:** By enabling interactions across broader regions through subsequent layers, it allows the model to effectively learn both local features and global image contexts.
- **Scalability:** This approach scales linearly rather than quadratically with image size, making it feasible for high-resolution image processing.

D. Training Procedure and Memory Management

Agent Configuration:

- Utilizes a replay buffer to store experiences, allowing the agent to learn from a diverse set of past actions, which is critical for effective learning.
- Employs a sophisticated experience replay mechanism that randomly samples previous experiences to update network weights, thus breaking the correlation between consecutive learning steps and enhancing learning stability.

Exploration Strategy:

- An epsilon-greedy strategy is used to balance exploration and exploitation. The agent chooses random actions with a probability that decays over time, encouraging exploration in early phases and more exploitation of learned values as it becomes more experienced.

E. Experience Replay and Model Updates

- **Experience Replay:** Periodically, the agent updates its knowledge by sampling a batch from its memory and learning from it, using the differences between predicted and actual outcomes to reduce prediction errors.
- **Model Synchronization:** The weights of the predicted Q-network are occasionally copied to the target Q-network to stabilize learning, preventing drastic shifts in the learning targets during training.

F. Replicability and Validity

The detailed specification of the agent's architecture, combined with the explicit training and evaluation procedures, ensures that the study is replicable. The use of common libraries and frameworks further supports the reproducibility of the research. The methodology outlined provides sufficient detail for others to recreate the study, evaluate its findings, and extend the work if desired.

IV. Repositories and Contributions

OpenAI Gym

Repository: https://github.com/openai/gym

What We Used: We utilized the "SuperMarioBros-v0" environment from OpenAI Gym as the primary simulation platform for our AI. This environment is standard for testing AI models in game scenarios.

Our Unique Contribution: We streamlined the default action space from 256 discrete actions to just 5. This significant reduction was strategic, aiming to enhance our model's decision-making efficiency and enable faster learning without the complexity of unnecessary actions. This change is crucial for practical application where time and computational resources are limited..

Training DQN to Play Super Mario Bros

Repository: https://github.com/GuanyaShi/Training-DQN-to-play-Super-Mario-Bros

What We Used: This repository provided a solid foundation on how to implement the Deep Q-Network (DQN) for playing Mario, which includes basic strategies for training reinforcement learning models.

Our Unique Contribution: We implemented the Double Deep Q-Network (DDQN), enhancing our model's learning process's stability and accuracy by more precisely approximating action values.

Mario Reinforcement Learning

Repository: https://github.com/DrJessop/MarioReinforcementLearning

What We Used: We adapted the environment wrapping techniques from this repository, which are essential for modifying the game's input states to suit our AI's learning structure.

Our Unique Contribution: We advanced the basic DDQN framework by incorporating a hybrid neural network architecture that combines ResNet-18 and SWIN Transformer. This approach goes beyond the simple CNN used in the repository.

By employing the SWIN Transformer, we introduced a way to better capture the hierarchical nature of the game's environment through attention mechanisms, which are particularly adept at handling the spatial and sequential complexities of "Super Mario Bros."

A. Innovations and Outcomes

- **Advanced Neural Architecture:** By combining ResNet-18 and SWIN Transformer, our model can process and learn from the game's spatial and temporal dynamics effectively.
- **Demonstrated Superiority:** Our model demonstrates markedly better performance, achieving higher scores and demonstrating more advanced strategic behaviors than traditional models.

V. Results

A. Performance Comparison

The performance of the AI models was evaluated based on the average rewards obtained over episodes. The following figures show the comparison between the generic CNN model and our proposed DDQN model with SWIN Transformer.

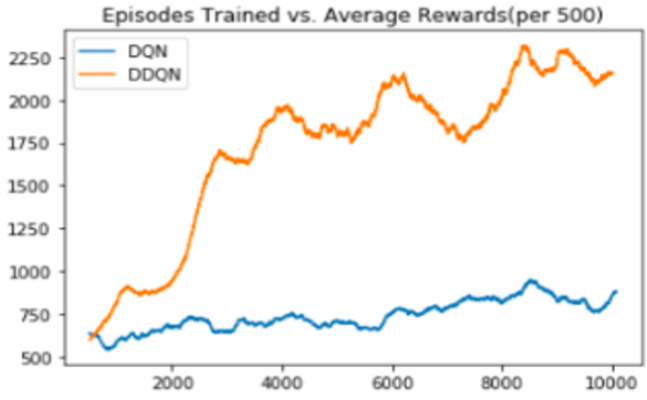

Fig. 4. Average Rewards for Generic CNN Model

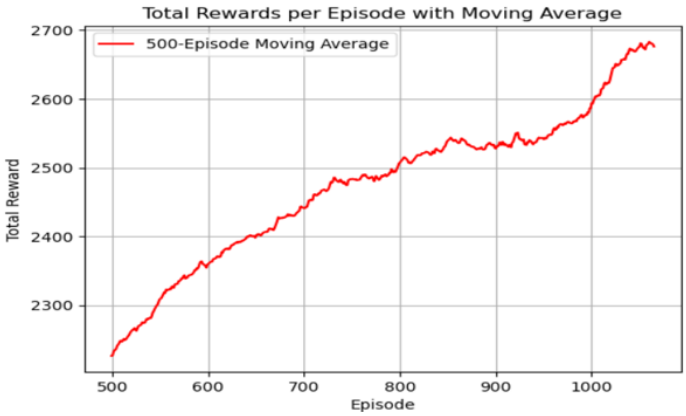

Fig. 5. Average Rewards for Our Model (DDQN with SWIN Transformer)

B. Analysis of Results

a) Generic CNN Model with DDQN:: Figure 4 shows the performance of the generic CNN model. The rewards start low and show a gradual increase over the episodes, indicating that the model struggles to improve significantly as training progresses. The generic CNN model takes approximately 10,000 episodes to reach an average reward of 2,250.

b) Proposed Model with DDQN and SWIN Transformer:: Figure 5 illustrates the performance of our proposed model. In contrast to the generic CNN model, our model shows a much steeper increase in average rewards early in the training process. The rewards continue to improve steadily, demonstrating the model's ability to learn and adapt more effectively over time. Remarkably, our model reaches an average reward of 2,700 consistently in just 1,250 episodes, significantly outperforming the generic CNN model both in terms of speed and final reward values.

c) Comparative Insights:: The comparison clearly indicates the superiority of our proposed model with the DDQN and SWIN Transformer architecture. The introduction of the SWIN Transformer enhances the model's ability to process spatial information efficiently, leading to better performance and higher rewards. The use of the shifted window mechanism within the SWIN Transformer likely contributes to this improvement by allowing the model to focus on relevant parts of the input data more effectively. The results suggest that the proposed model not only learns faster but also achieves higher performance levels compared to the generic CNN model.

C. Simulation and Testing

Video Demonstration: The effectiveness and advanced capabilities of our AI are demonstrated in this simulation video: https://drive.google.com/file/d/1sDmT4OxLW16u8LjSfMeDprTD4lfKdk6R/view?usp=sharing

VI. Conclusion

This study presents a comprehensive exploration of integrating Double Deep Q-Network (DDQN) with a SWIN Transformer to develop an AI agent capable of playing "Super Mario Bros" autonomously. The results highlight significant advancements in performance and learning efficiency over traditional models.

A. Key Findings

The key findings of the study are as follows:

- The proposed model, combining DDQN with the SWIN Transformer, significantly outperforms the generic CNN model. It achieved higher average rewards and demonstrated a steeper learning curve.
- Our model reached an average reward of 2,700 consistently in just 1,250 episodes, whereas the generic CNN model required approximately 10,000 episodes to reach an average reward of 2,250.
- The introduction of the SWIN Transformer, with its shifted window mechanism, allowed the model to process

spatial information more effectively, leading to better decision-making and faster learning.

B. Implications

These findings have several important implications:

- The integration of advanced neural architectures like the SWIN Transformer in reinforcement learning models can substantially enhance their performance, especially in complex environments requiring sophisticated spatial awareness.
- The efficiency and effectiveness of the proposed model suggest that similar approaches could be applied to other games and real-world applications where visual processing and quick decision-making are crucial.

C. Contributions to the Field

This project contributes to the field of AI and reinforcement learning in several ways:

- It demonstrates the practical benefits of combining DDQN with advanced neural network architectures for improving the performance of AI agents in gaming environments.
- The study provides a detailed methodology and analysis that can serve as a reference for future research and applications involving complex visual tasks and reinforcement learning.

D. Limitations

While the results are promising, there are limitations to consider:

- The study focuses on a single game environment, and the performance of the proposed model may vary in different contexts or with different types of games.

E. Future Research

Future research could explore several avenues:

- Applying the proposed model to a variety of games and real-world scenarios to assess its generalizability and robustness.
- Investigating the impact of different hyperparameters and architectural modifications on the performance of the model.

In conclusion, the integration of DDQN with a SWIN Transformer represents a significant step forward in the development of intelligent agents capable of mastering complex environments. This study not only showcases the potential of advanced neural architectures in reinforcement learning but also provides a foundation for future explorations and applications in the field.

VII. References

- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*.
- Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-learning. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (AAAI-16).
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *Proceedings of the 33rd International Conference on Machine Learning* (ICML-16).
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676), 354-359.
- Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv preprint arXiv:2103.14030*.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
- Levine, S., Pastor, P., Krizhevsky, A., & Quillen, D. (2018). Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *The International Journal of Robotics Research*, 37(4-5), 421-436.