RoboBounce: A Real-Time Pipeline for Stationary Ball-Hitting Tasks

Yu-Chung Chen	Yung-Shun Chan	Rong-Sheng Lin	Rong-Jyu Shih	Ting Yung Chen
R14944027	R14922102	R13922039	R13944025	B09902091
NTU, GINM	*NTU, CSIE*	*NTU, CSIE*	*NTU, GINM*	*NTU, CSIE*
yuzhong1214@gmail.com	assume0701@gmail.com	r13922039@ntu.edu.tw	ericpgz858@gmail.com	timonthy07@gmail.com

***Abstract*—This paper presents *RoboBounce*, an autonomous system designed for the task of continuous vertical ball-hitting using a 6-DOF robotic manipulator. The project integrates real-time computer vision, physics-based trajectory prediction, and a synchronized control framework built on ROS2 Humble. To overcome the challenges of high-speed tracking without extensive manual labeling, we utilize a teacher-student model architecture employing Grounding DINO for automated pseudo-labeling and YOLO for real-time inference. With the use of One Euro Filter, we successfully reduced the coordinate error margin from 10cm to 1cm, enabling high-precision tracking of the table tennis ball in the world frame. Despite achieving high spatial accuracy, the system encountered significant hardware-induced latency of approximately 200ms, which limited sustained performance to roughly three consecutive bounces. This report details the system architecture, the vision-based perception pipeline, and the control strategies utilized to bridge the gap between computer vision and robotic manipulation. The demonstration video for RoboBounce is available at https://youtu.be/l4kwCzyMjKc?si=bLbrcAAQoOqIPKye. The source code and implementation details for the UR5 robotic system are hosted on GitHub at https://github.com/YuZhong-Chen/RoboBounce.**

I. Introduction

The field of sports robotics has long served as a benchmark for evaluating the limits of real-time perception and high-speed manipulation. Table tennis, in particular, requires a seamless integration of rapid object detection, precise trajectory estimation, and low-latency motor control. Our team's motivation for developing *RoboBounce* stems from a shared passion for table tennis and a desire to tackle the inherent complexity of stationary ball-hitting tasks. This project provided a unique pedagogical opportunity to synthesize our proficiencies in computer vision with core robotics principles, such as kinematics, while exploring the challenges of real-time systems outside our typical comfort zones.

Current research in table tennis robotics often focuses on complex competitive scenarios, such as human-versus-robot matches or high-speed rallying. While these systems demonstrate the pinnacle of robotic agility, they often rely on expensive multi-camera setups and high-end industrial controllers in combination with end-to-end machine learning models. In contrast, our objective was to implement a robust vertical bouncing pipeline using a single Intel RealSense D435i camera and a Universal Robots UR5 arm.The UR5 was chosen due to its superior servo quality and accessibility through a group member's laboratory, providing a high-performance and efficient platform for testing our object detection and control algorithms.

The *RoboBounce* pipeline is structured across five critical layers: Perception, Detection, Estimation, Decision, and Control. A central challenge in this implementation was the conversion of 2D image data into actionable 3D world-frame coordinates—a process requiring precise Eye-to-Hand calibration. Furthermore, the noise inherent in depth sensing necessitated a robust filtering strategy. By adopting the One Euro Filter, we were able to minimize jitter and reduce the robot arm's motion error margin from 10cm to 1cm, significantly increasing the reliability of the "Strike" phase.

However, the primary bottleneck encountered during the project was system-wide latency. The delay between the perception of the ball and the actual execution of the robot's movement after issuing the command reached approximately 200ms. In a sport where milliseconds determine success, this latency proved to be a formidable hurdle. While the tracking and command logic remained precise, the temporal gap caused the robot to consistently lag behind the ball's optimal hitting window. We attempted issuing the command to account for the latency, however the current control scheme still disallows the robot arm to strike the ball in time. Consequently, while our ideal metric was indefinite continuous bouncing, the current hardware constraints limited our results to approximately three successful cycles. Despite this, the project demonstrates a successful integration of deep learning-based detection with classical robotic control, providing a foundation for future optimizations in latency-compensated motion planning.

II. Related Work

Robotic table tennis has been widely studied as a benchmark problem for real-time perception, trajectory prediction, and control. Existing works can be broadly categorized into learning-based control approaches, physics-based trajectory modeling, and vision-based ball tracking with temporal filtering. This section reviews prior studies along these three directions and highlights their respective limitations.

A. Learning-Based Approaches for Robotic Table Tennis

Learning-based methods have been extensively explored in robotic table tennis to address the highly nonlinear dynamics

of ball motion and racket–ball interaction. Early work demonstrated that robotic systems can acquire striking behaviors through data-driven approaches rather than explicit analytical modeling. Muelling *et al.* [1]–[3] presented a representative learning-based robotic table tennis system in which an anthropomorphic robotic arm learned human-like stroke motions through imitation learning. Their system integrated perception, trajectory prediction, and motor control into a unified learning framework, showing that learning-based approaches can achieve adaptive behavior in dynamic environments.

To improve interpretability and data efficiency, later studies adopted modular learning architectures instead of fully end-to-end models. Matsushima *et al.* [4], [5] decomposed the table tennis task into multiple input–output mappings, including the relationship between incoming ball states, racket motion, post-impact ball velocity, and landing position. These mappings were learned using Locally Weighted Regression (LWR), enabling the system to capture nonlinear behaviors while preserving a structured pipeline.

Building upon this framework, Huang *et al.* [6] combined LWR with fuzzy cerebellar model articulation control to incorporate adaptive feedback based on observed landing errors. Although this approach improved robustness through iterative refinement, it relied on repeated trial-and-error learning and was therefore less suitable for scenarios requiring accurate prediction within a short observation window. To reduce the computational overhead of LWR, Zhang *et al.* [7] replaced LWR with neural networks to learn post-impact ball velocity and racket parameter mappings, achieving faster inference with fixed model complexity.

Despite the success of learning-based approaches, fully end-to-end formulations are not always practical for real-time robotic table tennis systems. End-to-end models typically require large-scale annotated datasets covering diverse ball trajectories, spin conditions, and environments, which are costly and time-consuming to collect. Moreover, such models lack explicit physical interpretability, making failure diagnosis and robustness analysis difficult under sensing noise or unexpected motion. In addition, deep end-to-end pipelines often introduce non-negligible inference latency, which can challenge real-time execution at high frame rates. These limitations motivate the use of structured and modular designs that incorporate prior knowledge while maintaining computational efficiency.

B. Physics-Based Trajectory Modeling and Control

Physics-based approaches model the motion of a table tennis ball by explicitly considering gravity, aerodynamic drag, the Magnus effect induced by spin, and elastic collisions with the table and racket. Nakashima *et al.* [8]–[10] developed comprehensive physical models that combine aerodynamic motion equations with rebound models for ball–racket interaction. These formulations enable accurate forward simulation of ball trajectories when physical parameters are known.

However, controlling the landing position of the ball requires solving an inverse problem, in which the desired landing point must be mapped back to appropriate racket orientation, velocity, and timing. This inverse operation is highly nonlinear and sensitive to environmental parameters such as air resistance coefficients and restitution factors, which are difficult to measure precisely in practice. Consequently, many physics-based methods simplify the problem by assuming limited spin conditions or neglecting rotational dynamics, thereby restricting their applicability to real-world scenarios involving diverse ball behaviors.

C. Ball Tracking and Temporal Filtering

Reliable ball tracking is a fundamental requirement for both learning-based and physics-based robotic table tennis systems. With the advancement of deep learning, object detection models have become the dominant solution for localizing fast-moving table tennis balls under varying lighting conditions and backgrounds [11]. However, constructing large-scale annotated datasets for ball detection remains challenging due to the small size of the ball, motion blur, and frequent occlusions.

To reduce annotation cost, recent studies have explored weakly supervised or semi-automatic dataset generation strategies. Vision–language grounding models enable object localization based on textual prompts, allowing bounding box annotations to be generated automatically with minimal human intervention [12]. Such approaches provide a scalable alternative for dataset construction and are often combined with fine-tuning lightweight detectors, such as YOLO-based models, for real-time inference [11].

Even with robust detectors, visual measurements of ball position are inherently noisy. Temporal filtering is therefore widely used to improve state estimation stability. Kalman filtering, as introduced in *An Elementary Introduction to Kalman Filtering*, has been extensively applied to smooth ball trajectories and estimate velocity by modeling motion dynamics under Gaussian noise assumptions [13]. In addition, adaptive smoothing techniques such as the One Euro Filter, originally proposed for noisy input in interactive systems, have been adopted to balance noise suppression and responsiveness [14]. These temporal filtering methods play a crucial role in bridging noisy visual observations and reliable trajectory prediction in real-time robotic table tennis systems.

III. Methodology

The *RoboBounce* system is an integrated robotic framework designed for autonomous and continuous table tennis ball control. The methodology combines high-speed computer vision, physics-based trajectory modeling, and robust motion control to achieve stable vertical bouncing.

A. System Architecture and Hardware Setup

The hardware foundation consists of a 6-DOF Universal Robots UR5 industrial arm, which provides the necessary reach and payload capacity for high-speed striking tasks. To facilitate the interaction with the ball, a specialized end-effector assembly was developed.

Perception is handled by an Intel RealSense D435i camera providing synchronized RGB-D streams. The software architecture is built upon ROS2 Humble, employing a distributed

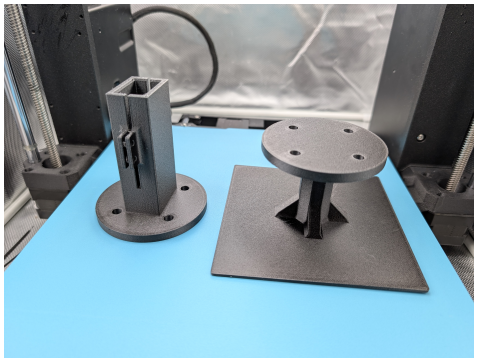

Fig. 1. Custom 3D-printed end-effector adapter designed for the UR5 flange.

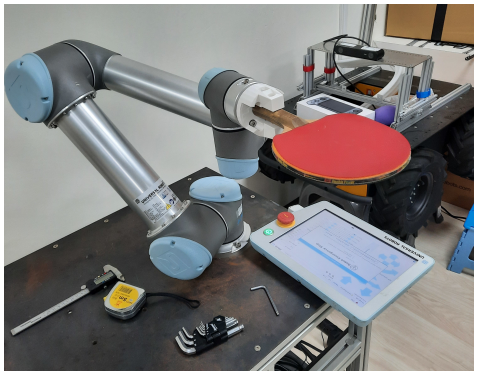

Fig. 2. The integrated table tennis racket assembly mounted on the robot.

node system to manage data flow between the vision pipeline and the UR robot driver.

B. Vision-Based Perception Pipeline

The vision system is required to estimate the 3D position of a fast-moving ball with minimal latency. Our pipeline utilizes deep learning for detection and statistical methods for depth refinement.

1) Object Detection and Automated Labeling: To balance accuracy and real-time performance, we utilize a teacher-student training paradigm. **Grounding DINO** serves as a zero-shot teacher model to automatically generate pseudo-labels from video datasets of the ball in motion. These labels, after manual filtering, are used to fine-tune a **YOLO**-based student model.

2) Robust Depth Estimation: Raw depth data from RGB-D sensors often contains noise, especially at the edges of small moving objects. We implement a refinement strategy to ensure spatial accuracy:

- **Bounding Box Shrinkage**: The 2D bounding box is reduced by 50% towards its center to ensure only pixels belonging to the ball are sampled.
- **Statistical Filtering**: We calculate the average of the first 60% of depth values within the refined box to discard noisy outliers and background artifacts.

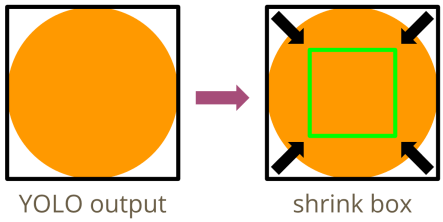

Fig. 3. Visualization of the bounding box shrinkage technique to filter out edge noise.

3) Eye-to-Hand Spatial Calibration: To map coordinates from the camera frame to the robot's base frame ($base\_link$), an eye-to-hand calibration is performed.

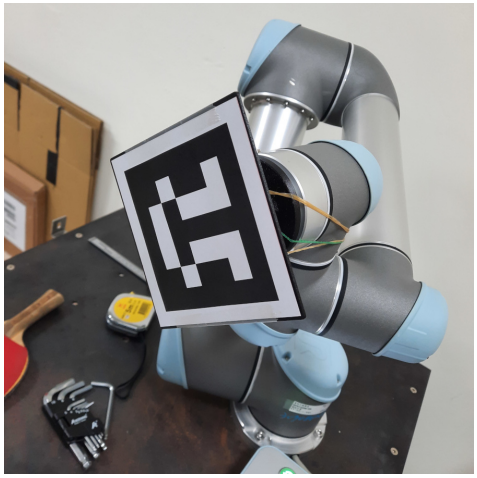

Fig. 4. ArUco marker plate used as a spatial reference for calibration.

Fig. 5. Visualization of the coordinate transformation and detected marker in the environment.

By capturing 15–20 samples of an ArUco marker at various robot poses, we estimate the static transformation matrix. This allows the system to publish ball positions directly within the robot's operational workspace.

C. Trajectory Prediction and State Management

Predictive modeling is essential to compensate for the time elapsed between perception and physical contact.

1) Physics-Based Trajectory Prediction: The system estimates the ball's initial velocity vector by analyzing the first five frames of its motion. A constant-acceleration model is then applied to predict the future intercept point. To smooth the data and reduce jitter, we adopt the **One Euro Filter**.

2) State Machine Control: The robot's behavior is governed by a synchronous state machine:

1) **Idle**: The robot stays at a neutral home pose.
2) **Strike**: The system executes an upward strike based on prediction.
3) **Recover**: The robot returns to the home pose to prepare for the next bounce.

D. Implementation Challenges and Analysis

Real-world deployment revealed significant control and timing issues that necessitated further optimization.

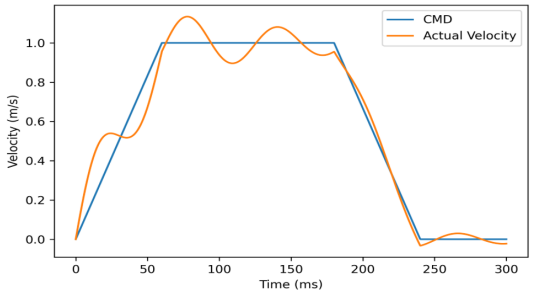

Fig. 6. Analysis of velocity overshoot encountered during initial PID tuning.

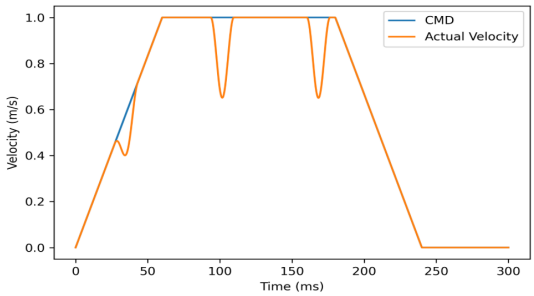

Fig. 7. Discrepancies between commanded and actual velocity due to driver timeouts.

1) Latency Compensation: The system encountered a latency of approximately 200 ms between the computing unit and the UR5 controller. This delay results in a vertical error $\Delta h \approx 0.196$ m if left uncompensated.

To address this, we integrated the latency directly into the trajectory prediction, commanding the robot to strike where the ball is estimated to be 200 ms in the future.

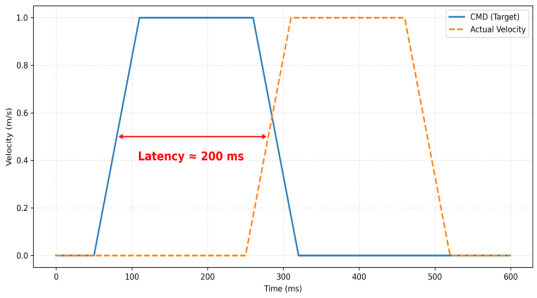

Fig. 8. Latency analysis showing the delay between the target command (CMD) and actual motor execution.

IV. Experiment

A. Experiment Setup

In this section, we present the experimental setup and evaluate the performance of the RoboBounce system in a real-world environment.

B. Hardware Configuration

The experimental platform is constructed to facilitate high-speed interaction with a table tennis ball. The core components are:

- **Robotic Manipulator**: A Universal Robots UR5 6-DOF industrial arm is utilized for its high repeatability and ROS2 support.
- **End-Effector**: A standard table tennis racket is mounted to the UR5 tool flange using a custom 3D-printed adapter.
- **Vision Sensor**: An Intel RealSense camera provides synchronized RGB and depth streams, fixed in an eye-to-hand configuration.
- **Calibration Tools**: A custom-designed ArUco calibration board is utilized for high-precision spatial transformation between the camera and the robot base_link.
- **Computing Unit**: All processing, including YOLO-based detection and trajectory prediction, is executed on a remote PC running ROS2 Humble.

C. Calibration Strategy and Accuracy

A critical challenge in the setup was achieving sufficient hand-eye calibration accuracy. Initially, we attached a single ArUco marker directly to the robotic arm's end-effector. However, this method yielded a significant spatial error of approximately ± 10 cm, which was inadequate for hitting a table tennis ball.

To improve precision, we developed a dedicated calibration board and temporarily increased the RealSense RGB resolution to 1920×1080. This higher pixel density allowed for more precise corner detection, successfully reducing the calibration error to within ± 1 cm.

While the 1920×1080 resolution provides high precision, it is limited to 8 FPS and is incompatible with the depth sensor's alignment at that scale. Therefore, for the real-time tracking pipeline, we switch the resolution to 424×240 to achieve a 60 FPS update rate. This high frame rate is essential for the

YOLO-based detection pipeline to capture the high-velocity motion of the ball.

D. System Performance and Stability

To evaluate the stability of the closed-loop perception and control system, we conducted multiple trials of vertical ball bouncing. The primary metric for success is the average number of continuous bounces.

The experimental results indicate that the system can maintain stable control despite a latency of approximately 200 ms. Table I summarizes the performance data.

TABLE I
ROBOBOUNCE PERFORMANCE EVALUATION

Metric	Value
Robot Platform	UR5 Arm
Detection Model	Fine-tuned YOLO
Calibration Accuracy	$< \pm 1$ cm
Real-time Update Rate	60 FPS
Average Bounces	**4.3**

E. Discussion of Error Factors

The performance of the RoboBounce system was primarily limited by the following factors:

- **Depth Mapping Noise**: To maintain the 60 FPS update rate required for high-speed tracking, we utilized a lower-resolution depth stream. This resulted in higher noise levels and less accurate depth mapping of the fast-moving ball, leading to errors in the Z-axis trajectory prediction.
- **Kinematic Constraints of the Manipulator**: A significant hardware bottleneck was the **limited acceleration of the UR5 arm**. In many instances, although the system correctly predicted the hitting point, the arm's joints could not accelerate quickly enough to reach the designated spatial coordinates and match the required striking velocity within the millisecond-level window.

Future work will focus on integrating optical flow to improve trajectory estimation and developing motion planning algorithms that better compensate for the hardware's acceleration limits.

V. CONCLUSIONS

In this project, we presented *RoboBounce*, a robotic system capable of autonomous and continuous table tennis ball control using a UR5 manipulator. By integrating a real-time vision pipeline with physics-based trajectory prediction, the system effectively bridges the gap between perception and high-speed robotic action.

Our technical contributions include a robust detection framework that leverages Grounding DINO for automated pseudo-labeling to fine-tune a high-performance YOLO model. Furthermore, we implemented a depth-filtering strategy and an Eye-to-Hand calibration process using ArUco markers to ensure accurate spatial mapping from the RealSense camera to the robot's workspace. To address the inherent 200 ms latency between the remote PC and the UR controller, we employed a constant-acceleration motion model and a One Euro Filter for smooth and predictive striking.

Experimental results demonstrate that the system achieves an average of 4.3 continuous bounces. While the current implementation proves the feasibility of the architecture, performance remains limited by depth sensor noise and control delays.

Future work will focus on:

- Improving trajectory estimation by incorporating optical flow and multi-view camera setups.
- Exploring advanced depth-prediction models, such as Depth-Anything, to replace raw depth maps for higher precision.
- Optimizing the control loop to mitigate PID overshoot and driver-related timing issues.

Through these enhancements, we aim to achieve more human-like, long-term stability in robotic ball control tasks.

VI. WORK DIVISION

The division of labor within our team is summarized in Table II. Yu-Chung Chen was primarily responsible for the hardware assembly of the UR robotic arm and the configuration of the RealSense camera and other peripherals. Yung-Shun Chan focused on implementing object tracking and pose filtering to enhance estimation accuracy. Together, they developed the ROS-based system and the overall pipeline, including the state machine for core control logic. The other three members, Rong-Sheng Lin, Rong-Jyu Shih and Ting-Yung Chen, assisted with data labeling using Grounding DINO, fine-tuning the YOLO model, and designing the custom fixture for the table tennis racket.

TABLE II
WORK DIVISION IN DETAIL.

Name	Work
Yu-Chung Chen	UR / Realsense Driver Setup Pipeline Implementation / Deployment
Yung-Shun Chan	Object Tracking / Pose Filtering Pipeline Implementation / Deployment
Rong-Sheng Lin	Fine-tuning YOLO Design and print the 3D prints
Rong-Jyu Shih	Fine-tuning YOLO Design and print the 3D prints
Ting-Yung Chen	Fine-tuning YOLO Design and print the 3D prints

REFERENCES

[1] K. Mülling, J. Kober, and J. Peters, "Learning table tennis with a mixture of motor primitives," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2010, pp. 411–416.

[2] K. Mülling, J. Kober, O. Krömer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 263–279, 2012.

[3] K. Mülling, J. Kober, and J. Peters, "A biomimetic approach to robot table tennis," *Adaptive Behavior*, vol. 18, no. 5, pp. 359–376, 2010.
[4] M. Matsushima, T. Hashimoto, and F. Miyazaki, "Learning to the robot table tennis task: Ball control and rally with a human," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2003, pp. 2962–2969.
[5] M. Matsushima, T. Hashimoto, M. Takeuchi, and F. Miyazaki, "A learning approach to robotic table tennis," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 767–771, 2005.
[6] Y. Huang, D. Xu, M. Tan, and H. Su, "Adding active learning to lwr for ping-pong playing robot," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1489–1494, 2012.
[7] K. Zhang, Z. Fang, J. Liu, and M. Tan, "Modeling the stroke process in table tennis robot using neural network," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 243–248.
[8] A. Nakashima, Y. Ogawa, Y. Kobayashi, and Y. Hayakawa, "Modeling of rebound phenomenon of a rigid ball with friction and elastic effects," in *Proceedings of the American Control Conference (ACC)*. IEEE, 2010, pp. 1410–1415.
[9] A. Nakashima, J. Nonomura, C. Liu, and Y. Hayakawa, "Hitting backspin balls by robotic table tennis system based on physical models of ball motion," in *IFAC Proceedings Volumes (10th IFAC Symposium on Robot Control)*, vol. 45, no. 22, 2012, pp. 834–841.
[10] A. Nakashima, D. Ito, and Y. Hayakawa, "An online trajectory planning of struck ball with spin by table tennis robot," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2014, pp. 865–870.
[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[12] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, J. Zhu, and L. Zhang, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," 2024.
[13] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Tech. Rep., 1995, technical Report.
[14] G. Casiez, N. Roussel, and D. Vogel, "The 1 filter: A simple speed-based low-pass filter for noisy input in interactive systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2012, pp. 2527–2530.