Robot ajedrecista: sistema mecatrónico aplicado a la toma de decisiones

Guillermo Larregay[1], Federico Pinna[1], Luis Avila[12], y Daniel Morán[1]

[1] Laboratorio de Mecatrónica, FICA-UNSL, Ruta Prov. 55 Ext. Norte, San Luis D5730EKQ, Argentina
[2] Laboratorio de Investigación y Desarrollo en Inteligencia Computacional, FCMyN-UNSL, Av. Ejército de Los Andes 950, San Luis D5700HHW, Argentina
{golarregay,fpinna,loavila,dmoran}@unsl.edu.ar

Resumen Este trabajo presenta un sistema mecatrónico compuesto por un brazo robótico que juega ajedrez de forma autónoma. El sistema se basa en un robot manipulador de tipo industrial, un sistema de visión artificial, y un motor de juego basado en software libre. La robótica aplicada a juegos interactivos constituye un excelente problema para explorar la colaboración humana-robot, ya que presenta una estructura cuya complejidad puede ser gradualmente incrementada. Por lo tanto, este desarrollo debe entenderse como una primera aplicación exitosa de un sistema mecatrónico de toma de decisiones factible de migrar a otros usos y contextos.

Palabras claves: mecatrónica, toma de decisiones, visión artificial, ajedrez.

1. Introducción

Un sistema mecatrónico constituye un sistema controlado que integra herramientas de software junto a componentes eléctricos, electrónicos y mecánicos [3]. En este tipo de sistemas se emplean sensores para percibir el entorno y microprocesadores para las tareas de procesamiento, que derivarán posteriormente en acciones de control que los actuadores transformarán en los cambios físicos deseados. De esta manera, en la mecatrónica se conjuntan tecnologías relacionadas a los sistemas de percepción del entorno, sistemas de accionamientos, análisis del comportamiento dinámico de sistemas, sistemas de control, sistemas de procesamiento, etc.

Este trabajo presenta un sistema mecatrónico compuesto por un brazo robótico que juega ajedrez de forma autónoma con un oponente. Brevemente, la aplicación se basa en un robot manipulador de tipo industrial, un sistema de visión artificial, y un motor de juego basado en software libre. De manera general, los juegos de mesa implican la percepción e identificación de un conjunto de piezas en un tablero, el conocimiento de las reglas o posibles movimientos a ejecutar, representación del estado actual del juego para el posterior razonamiento o toma de decisiones, y la manipulación de las piezas físicas; todo esto mientras se

coordina con un oponente (humano o artificial). Aún cuando un gran numero de este tipo de autómatas han sido desarrollados con propósitos de entretenimiento, este puede considerarse como una fase de prueba que puede incorporar mayor complejidad para el estudio de sistemas de percepción y manipulación en entornos reales con ruido, incertidumbre y restricciones físicas [2]. De esta manera, cualquier avance en este tipo de sistemas autónomos aplicados a juegos de mesa, allana el camino para otros sistemas más específicos y complejos de cooperación humana-robot. Por ejemplo, un juego interactivo originariamente diseñado con fines didácticos, podría conducir finalmente al desarrollo de un manipulador capaz de servir de asistente de laboratorio en tareas que impliquen el manejo de sustancias nocivas. Específicamente, un gran número de trabajos que describen la interacción humana-robot, han analizado extensivamente el fenómeno de toma de decisiones por turnos en distintos contextos. Desde procesamiento del habla con dos o más participantes [4], a juegos de roles aplicados a la terapia de pacientes con autismo [5].

Existen en la literatura una variedad de trabajos que describen realizaciones de sistemas que involucran a robots en juegos de estrategia como el ajedrez y las damas entre otros. Estos desarrollos tienen por objetivo mostrar la capacidad de integración de diferentes tecnologías como la robótica, la visión artificial y la inteligencia artificial. Entre estas realizaciones, el Marine-Blue [9] es sin duda una de las más destacadas. Este sistema combina un robot manipulador, un sistema de visión y un motor de juego basado en el paquete de software Chessterfield. No obstante, los sistemas de juego de mesa autónomos utilizan tableros instrumentados y piezas especialmente co-diseñadas con el manipulador con la finalidad de simplificar las tareas de percepción y manipulación. Esto deriva en diseños específicos [7] que no contemplan en la fase de diseño las complejidades introducidas por el uso de piezas y tableros arbitrarios y entornos reales. No resulta entonces difícil comprender por qué la eficiencia de estos sistemas se degrada rápidamente cuando las especificaciones o parámetros de operación cambian.

El sistema mecatrónico presentado en este trabajo representa un salto adelante en la generalidad, lo que garantiza la factibilidad de ejecución de múltiples tareas. Particularmente, no requiere de una instrumentación especial o de un ajustado modelado de piezas o tableros con el fin de facilitar la visión y los movimientos. El tablero no esta fijo en relación al brazo del robot y como consecuencia se calibra de forma continua durante el juego. Mediante el sistema de visión se monitorea el estado del tablero de forma continua, lo que permite detectar cuándo y qué tipo de movimiento a ejecutado el oponente.

2. Descripción del sistema

La característica principal del sistema mecatrónico presentado es que su base está formada por un brazo robótico de tipo industrial, lo que contribuye a tener un impacto más generalizado en cuanto a la diversidad de tareas que podrían llevar a cabo. La utilización de un brazo robótico de tipo industrial permite además crear una plataforma abierta y flexible que soporte la futura expansión

Figura 1. Brazo robótico de 6 grados de libertad.

tanto en términos de hardware como software. En este sentido, el manipulador consta de un brazo marca ABB [3], modelo IRB120 de 6 grados de libertad (gdl). La disposición del brazo robótico puede observarse en la Fig. 1. Los tres primeros gdl son de rotación y proporcionan el control de posición, mientras que los últimos 3 proporcionan el control y orientación de la mano (extremo) del robot. La mano consiste en una pinza *gripper* de prensado paralelo con 4 dedos, y puede ser fácilmente reemplazable para adaptarse a tareas especializadas donde se requiere otro tipo de dispositivo de agarre. Se utilizó una cámara genérica asociada a un conjunto de LEDs que garanticen una distribución de iluminación uniforme sobre las piezas de ajedrez. Tanto el soporte de la cámara y los LEDs como las piezas de ajedrez, han sido construidos mediante una impresora 3D de fabricación propia.

En lo que se refiere a la arquitectura del sistema, ésta consta básicamente de los siguientes subsistemas:

- Subsistema de visión (cámara).
- Motor de juego.
- Software de la unidad de control central (SUCC).
- software de interfaz de comunicación (SIC) con el controlador del robot.
- Brazo robótico.

[3] ABB (Asea Brown Boveri) es una corporación multinacional, cuya sede central queda en Zürich, Suiza. `http://www.abb.com`

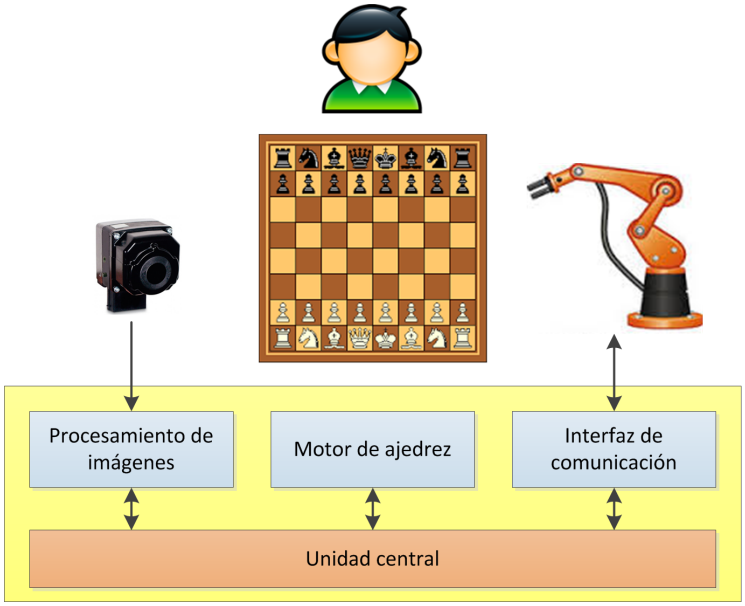


Figura 2. Diagrama de la estructura del robot ajedrecista.

Dicha arquitectura está organizada en torno a un módulo principal denominado *software de unidad de control central* (SUCC), tal como lo muestra la Fig. 2. El subsistema de visión se encarga de monitorear el estado del tablero, y debido a que éste no está fijo en relación con el robot su posición con respecto a la cámara, debe ser calibrada de forma continua durante el juego. Una vez que el sistema finaliza el reconocimiento de las piezas de ajedrez mediante el subsistema de visión, se evalúa el estado de la partida, se razona acerca del próximo movimiento a realizar y se envían instrucciones al mecanismo encargado de ejecutar las acciones correspondientes. Por último, el manipulador industrial mueve y/o captura las piezas de ajedrez involucradas. Posteriormente, el sistema da lugar al contrincante. Cuando el usuario está en posesión de su turno, el robot se encuentra en espera en su posición de reposo. El jugador cede el turno al robot mediante la activación de un botón asociado al temporizador de juego.

2.1. La visión del robot

Adquisición y procesamiento gráfico de la imagen. Con el fin de permitir el correcto reconocimiento de las piezas sobre el tablero, en primer lugar se toma una fotografía del mismo, para luego procesar la imagen obtenida de forma que se destaquen las características más importantes de las casillas y las piezas ubicadas en ellas. Con este fin, se ha utilizado una cámara de baja resolución, montada

junto al gripper, tal como se ve en la figura 3. Para el procesamiento de la imagen obtenida se utilizó la librería de código abierto OpenCV [4].

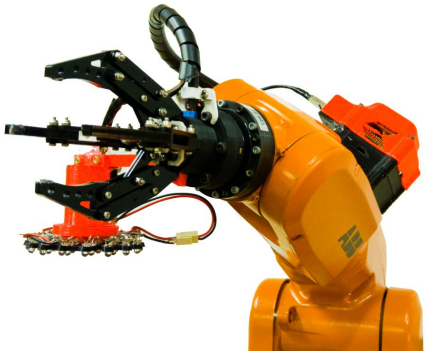
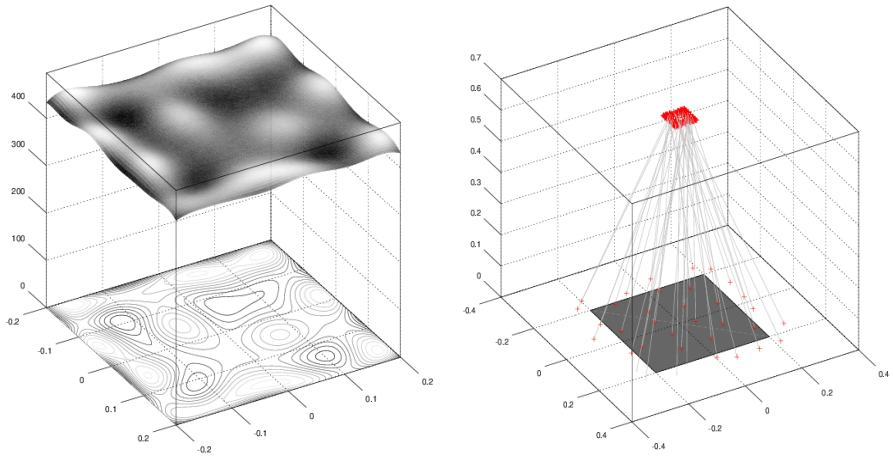


Figura 3. Detalle del gripper del robot con la cámara montada.

En el inicio del juego, el manipulador se mueve a una posición que permita que la totalidad del tablero sea visible para la cámara. Debido a que la proporción ancho/alto de la imagen capturada es de 4:3, se debe recortar la fotografía para visualizar solamente la sección correspondiente al tablero. Además, debido a las características cromáticas de la iluminación, es necesario corregir la temperatura de color de la imagen obtenida e incrementar la saturación de los colores posteriormente.

Antes de comenzar con el análisis de la imagen, ésta se procesada para evitar que las variaciones de iluminación sobre la superficie del tablero afecten a la detección de las piezas. Para ello, se implementó un algoritmo de ecualización de histograma por sectores, denominado ecualización de histograma adaptativo limitado por contraste (CLAHE, por sus siglas en inglés) [8]. Este algoritmo divide la imagen en sectores y transforma posteriormente el contraste de cada pixel en función de su entorno, de tal forma de lograr un rango de contraste máximo en cada sector de la imagen. La correcta distribución de iluminación sobre la superficie del tablero se lleva a cabo por un conjunto de diodos LEDs, cuya configuración se obtuvo mediante un algoritmo de tipo genético. Puede observarse en la Fig.4 la distribución de iluminación óptima obtenida, mientras que la metodología empleada se describe en [6]. Para aplicar la ecualización sobre las diferencias de iluminación, en primer lugar se debe convertir el espacio de color RGB de la imagen original a otro espacio de color que tenga un canal de luminancia, y luego aplicar CLAHE en este canal. Para el presente trabajo se utilizó el espacio de color L^*a^*b, ya que además de tener un canal de luminancia (L*), tiene la ventaja de que las distancias entre colores sobre las coordenadas

[4] OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel `http://opencv.org/`



(a) Niveles de iluminación sobre el tablero de ajedrez.

(b) Orientación de cada diodo LED.

Figura 4. Distribución de iluminación en el tablero de ajedrez.

a* y b* son proporcionales a las diferencias perceptuales entre colores. Una desventaja del algoritmo CLAHE es que en caso de existir un componente de ruido en la imagen, los resultados obtenidos no serán del todo óptimos. Una solución, es filtrar previamente la imagen utilizando desenfoque Gaussiano con el objetivo de eliminar el ruido térmico característico del sensor de la cámara y suavizando los colores de la misma.

Calibración de piezas y casillas vacías. Una vez obtenida la imagen recortada y mejorada, el sistema procede a detectar las diferentes piezas sobre el tablero. Para esto es necesario en primer lugar realizar una calibración, que consiste en determinar los valores y rangos de variación de los colores de las piezas blancas y negras, y de las casillas vacías claras y oscuras. Partiendo desde la posición inicial del tablero y antes de comenzar la partida, se puede asumir que las dos primeras filas del tablero contienen las piezas blancas (en el orden de inicio), y las dos últimas filas contienen las piezas negras. Por tal motivo, las casillas de color claro quedan ubicadas en las posiciones donde la suma de los índice de filas y de columnas es impar, mientras que las casillas oscuras están donde la suma es par. Esto deja una posición inicial con un total de 16 casillas con piezas blancas, 16 casillas con piezas negras, 16 casillas vacías de color claro y 16 casillas vacías de color oscuro. Tanto para la calibración como para la detección se toman siete muestras de color en cada casilla. En el caso del proceso de calibración, estos siete valores se promedian y se guarda este promedio como valor de referencia para la posterior comparación.

Detección de piezas. Para comprobar si una determinada casilla en el tablero está vacía u ocupada por una pieza, pueden evaluarse las coordenadas de los cuatro extremos de la casilla, o bien las coordenadas del centro de la misma y sus dimensiones (tanto ancho como alto). Por ello, se toman siete muestras de color para cada casilla: una muestra al centro de la misma, y las otras seis alrededor, formando los vértices de un hexágono de radio configurable por parámetros. El proceso de detección consiste en comparar cada una de las siete muestras de la casilla con los valores de referencia obtenidos durante la calibración del sistema. Debido a la posibilidad de que alguno de los puntos muestreados no caiga sobre la pieza, se implementó un algoritmo simple de votación de los resultados, que se resume a continuación:

- Si hay al menos dos muestras más cercanas al color de referencia de pieza blanca que a los de casilla vacía, y la cantidad de muestras cercanas a la referencia de pieza blanca es mayor a la cantidad de muestras cercanas a la referencia de pieza negra, considerar la casilla como ocupada por una pieza blanca.
- Si hay al menos dos muestras más cercanas al color de referencia de pieza negra que a los de casilla vacía, y la cantidad de muestras cercanas a la referencia de pieza negra es mayor a la cantidad de muestras cercanas a la referencia de pieza blanca, considerar la casilla como ocupada por una pieza negra.
- En cualquier otro caso, considerar la casilla como vacía.

Es importante destacar que el concepto de distancia entre colores utilizado es similar al de distancia euclidiana, tomando como ejes coordenados a los valores de R, G y B de cada pixel. Una vez analizadas todas las casillas, se crea una matriz de 8x8, donde cada elemento i, j de la misma es 1 si en la casilla i, j hay una pieza blanca, -1 si hay una pieza negra, y 0 si la casilla está vacía. Esta matriz es la que indica el estado actual del tablero, obtenido desde la fotografía.

Detección de movimientos e interacción con el motor de juego de ajedrez. Se utiliza como motor (back-end) para el juego de ajedrez al paquete de código abierto GNU Chess. Fue desarrollada una interfaz para interactuar desde el SUCC con el motor de ajedrez, con el fin de enviarle los movimientos realizados por el jugador humano, y hacer que el robot ejecute el movimiento devuelto por el motor de análisis. La detección del movimiento ejecutado por el jugador humano se lleva a cabo comparando dos matrices consecutivas de estados del tablero, como se observa en la Fig. 5. La primera contiene el estado previo al movimiento, y se obtiene luego de que el manipulador ejecute su jugada. La segunda matriz es la que se obtiene del análisis de la imagen obtenida con la cámara. Una ventaja de trabajar con matrices, es que se puede efectuar la operación de resta entre elementos para obtener los cambios entre la segunda y la primera, obteniendo así una matriz de diferencia como matriz de cambios. A continuación se muestra un ejemplo de detección para un movimiento normal.

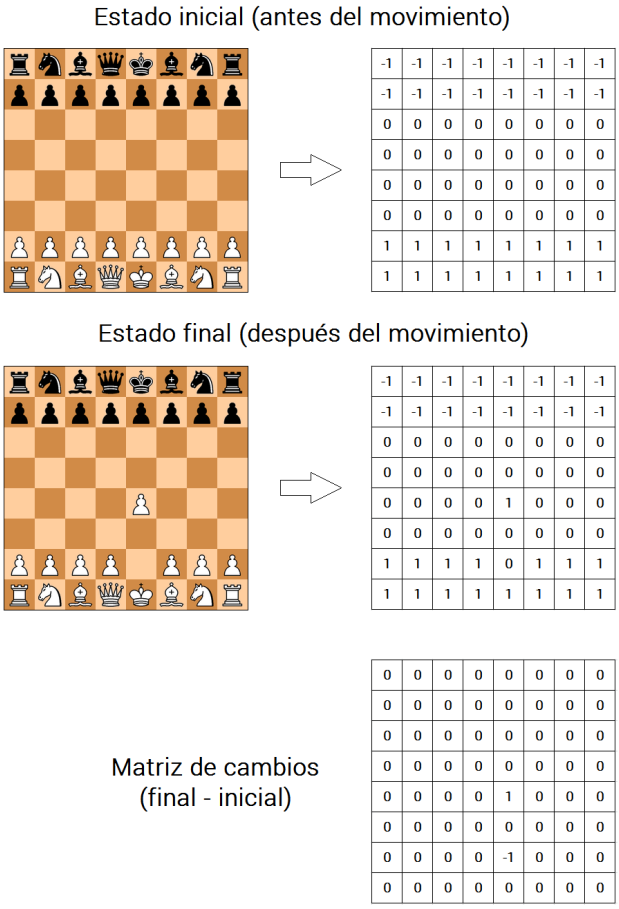


Figura 5. Obtención de la matriz de diferencias para un movimiento normal.

El sistema es capaz de detectar todos los movimientos legales que pueden darse en una partida de ajedrez, es decir: movimiento normal, captura, enroque corto y largo, coronación de peones y captura de peón al paso. Cada uno de estos movimientos tiene una "firma" particular dentro de la matriz de cambios evaluada para obtener el movimiento en notación de ajedrez. Es importante destacar que en la matriz de cambios, las casillas que no variaron se representan como ceros. Esta característica es útil para distinguir una firma particular. A continuación se analiza el comportamiento de la matriz de cambios para cada tipo de movimiento cuando el oponente juega con piezas blancas. Para el caso contrario, cambian los signos de la matriz. Además, puede verse en la Fig. 5 las matrices para cada uno de los casos.

- *Movimiento normal*: en este caso, si el jugador humano juega con piezas blancas, la casilla de origen del movimiento está representada por un valor -1, mientras que la de destino por un valor 1.
- *Movimiento de captura*: cuando el humano juega con piezas blancas el origen del movimiento es representado, igual que en el caso anterior, por un valor -1. Sin embargo, en este caso, la casilla de destino tiene valor 2.
- *Movimiento de enroque*: la detección del enroque es relativamente más simple, debido a que hay solamente dos posibilidades por color de piezas: enroque corto y enroque largo. Se analiza la primera o última fila del tablero (para el caso que el humano juegue en blancas o negras, respectivamente) y se compara con los valores conocidos $[-1\ 1\ 1\ -1]$ y $[-1\ 0\ 1\ 1\ -1]$ para enroque corto y largo respectivamente.
- *Movimiento de captura al paso*: este es un caso particular de captura, ya que la casilla de destino del peón que captura no es la de la pieza capturada. La casilla de origen del peón que captura está representada por un valor de -1 en el caso que el humano juegue con blancas. Sin embargo, debido a que el peón capturado no está en la casilla de destino, quedan dos casillas con valor 1 en la matriz de cambios. Esto se resuelve creando una matriz de cambios adicional, restando dos veces el valor de la matriz inicial al de la final. En esta nueva matriz, la casilla de destino del peón que capturó vale 1, mientras que la casilla del peón capturado vale 1 en la primera matriz de cambios y 2 en la segunda.
- *Movimiento de coronación de un peón*: esto ocurre cuando un peón blanco llega a la fila 8, o cuando un peón negro llega a la fila 1. Por defecto se considera que la coronación es a Dama, sin embargo se puede cambiar este comportamiento, para permitir la coronación a cualquier pieza válida.

2.2. Ejecución de acciones

Para que la unidad central SUCC pueda comunicarse con el controlador del robot, se desarrolló un *software de interfaz de comunicación* (SIC) entre el controlador del robot y la computadora embebida. El SIC corre bajo Windows y se programó en lenguaje C# con el entorno VisualStudio. Se eligió este lenguaje porque el fabricante provee una herramienta para comunicarse con el robot basada en el lenguaje ya mencionado [1]. Esta herramienta es conocida como PCSDK, la misma permite a los integradores de sistemas, terceros o usuarios finales añadir sus propias interfaces de usuario personalizadas para el controlador IRC5. Se pueden realizar aplicaciones de PC como independientes, que se comunican con el controlador del robot a través de una red. Esta herramienta utiliza Microsoft .NET y Microsoft Visual Studio.

El SIC inicia buscando controladores IRC5 en la red. Finalizada la búsqueda, el usuario deberá seleccionar el controlador para iniciar la comunicación con el mismo. Posteriormente, el sistema espera que el usuario configure el controlador en modo Automático y active los motores. Con esto ya configurado, se conecta el controlador en forma remota para acceder al programa y controlarlo desde el SUCC. Una vez conectado, se envían parámetros de inicialización y calibración

y se espera la señal del pulsador de fin de turno por parte del usuario. Una vez que el turno se activa, el SIC envía una señal al SUCC para que este comience a procesar la jugada. Una vez procesada, el SIC espera el paquete de datos (en formato XML) para enviarle una orden de ejecución de movimiento al brazo robótico. Los datos contenidos en el paquete se convierten en un grupo de símbolos que el controlador pueda procesar. Entre estos figuran: Tipo de jugada, coordenadas en el plano del tablero y alturas de las piezas que intervienen.

3. Resultados experimentales

Para comprobar el correcto funcionamiento del algoritmo de detección de piezas basado en fotografía, se jugaron 10 partidas completas contra el sistema y se contabilizaron los errores de detección en cada una. Se muestra el resultado de cada partida en el cuadro 1. Además, para establecer una comparación, se evaluaron los errores de detección en 10 partidas utilizando solamente la iluminación presente en el laboratorio (tubos fluorescentes, con iluminación no uniforme sobre la superficie de juego). Los resultados pueden verse en el cuadro 2.

Se considera un *falso positivo (FP)* si el sistema detecta una pieza (de cualquier color) en una casilla vacía, *falso negativo (FN)* si el sistema detecta una casilla vacía en lugar de una pieza, y *error de color (EC)* si detecta de forma incorrecta el color de una pieza. Se contabilizó la cantidad de movimientos efectuados por el sistema en cada partida. Para los casos donde se detectaron errores, un operador humano corrigió manualmente la matriz de detección, con el fin de continuar con la partida hasta el final.

Se puede ver en los cuadros 1 y 2 una mejora significativa en el comportamiento del sistema cuando se utiliza iluminación uniforme sobre el tablero. La tasa de error se ve reducida de un 11,76 % con la iluminación del laboratorio, a un 2,86 % con iluminación uniforme.

Partida	**Movimientos**	**FP**	**FN**	**EC**
1	24	1	0	0
2	31	0	0	0
3	19	0	0	0
4	22	0	0	1
5	21	2	0	0
6	37	0	0	0
7	23	1	0	1
8	19	0	0	0
9	21	0	0	0
10	28	1	0	0
Total	*245*	*5*	*0*	*2*

Cuadro 1. Resultados con iluminador LED encendido

Partida	Movimientos	FP	FN	EC
1	19	0	1	3
2	24	1	0	1
3	22	2	0	2
4	15	0	0	1
5	34	3	1	1
6	29	1	0	3
7	19	1	0	1
8	24	0	0	2
9	24	2	1	0
10	28	0	0	1
Total	*238*	*10*	*3*	*15*

Cuadro 2. Resultados con iluminador LED apagado

También se pueden observar ciertos comportamientos destacables. En primer lugar, cuando la iluminación no es uniforme, las sombras provocadas por las piezas sobre las casillas vacías o sobre piezas de menor altura afectan a la correcta detección del color o incluso de la presencia de una pieza en una casilla. Los falsos positivos son mayoritariamente casillas vacías de color oscuro que, cuando están parcialmente sombreadas, se confunden con piezas de color negro. Los falsos negativos en gran parte consisten en piezas blancas sombreadas, que se confunden con casillas vacías claras, mientras que los errores de color son en su totalidad piezas blancas sombreadas, en zonas de menor iluminación del tablero.

Cuando se tiene iluminación uniforme, se elimina completamente el error de falsos negativos, debido principalmente a que existe una mayor separación entre los umbrales de casilla vacía y de los colores de pieza. Sin embargo, debido a la ubicación de los LEDs con respecto a la cámara, en algunas piezas cuya parte superior es plana (por ejemplo: torre negra, caballo negro) se producen reflejos que el sistema considera como pieza blanca. Los falsos positivos, si bien son mucho menores, se mantienen notablemente altos. Es probable que esto sea así debido a que incluso con el iluminador uniforme funcionando, se encontraban encendidas las luces del laboratorio. Una posible solución a este problema es incrementar la cantidad de LEDs en el iluminador, de tal forma de lograr un mayor nivel de luz sobre el tablero, minimizando la incidencia y por ende, la generación de sombras causada por los tubos fluorescentes.

Se propone como trabajo a futuro cambiar el algoritmo de detección de piezas y calibración, con el fin de conseguir una mayor separación entre las clases. Esto eliminaría los errores presentes actualmente en el sistema, y permitiría explorar técnicas de agrupamiento que sean menos sensibles a los niveles de iluminación del tablero. Además, es posible también implementar algún algoritmo adaptativo que tenga en cuenta la cantidad de piezas de cada color en el tablero, y si detecta durante la partida un incremento en alguna de estas cantidades, cambie dinámicamente el umbral de detección. Se asume que en una partida de ajedrez normal, la cantidad de piezas de cada color en el tablero, entre jugadas disminuye o se mantiene constante. Esto eliminaría efectivamente el error de falsos positivos.

4. Discusión

La robótica aplicada a juegos interactivos constituye un excelente problema para explorar la colaboración humana-robot, ya que presenta una estructura cuya complejidad (al menos en términos de software) es fácil de ser incrementada. En este trabajo se presentó un sistema basado en un robot industrial capaz de jugar al ajedrez con un oponente humano de forma natural, utilizando piezas ordinarias y tableros no instrumentados. La interacción con el oponente humano se lleva a cabo mediante visión artificial, con cambios de turno mediante un simple botón y un temporizador de juego. La principal ventaja de este sistema mecatrónico frente a otros similares es que opera con una estructura general, sin requerir de un tablero de ajedrez instrumentado ni piezas especialmente diseñadas, por lo cual podrá fácilmente ser adaptado para realizar otras tareas.

A pesar de que el juego de ajedrez no es el objetivo final de nuestra investigación, esta constituye una aplicación exitosa de un sistema de manipulación factible de migrar a otros contextos. Este proyecto ha sido desarrollado en el Laboratorio de Mecatrónica de la Facultad de Ingeniería de la Universidad Nacional de San Luis y un video del sistema funcionando se pueden encontrar en `https://www.youtube.com/watch?v=cJwDVGww09Q`.

Referencias

1. ABB: Robot Communication Development, `http://developercenter.robotstudio.com:80/Index.aspx?DevCenter=RobotCommunication&OpenDocument&Url=html/`
2. Aliane, N., Bemposta, S.: Checkers playing robot: A didactic project. Latin America Transactions, IEEE 9(5), 821–826 (2011)
3. Bolton, W.: Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering. Essex. Pearson (2011)
4. Breazeal, C., Scassellati, B.: Infant-like social interactions between a robot and a human caregiver. Adaptive Behavior 8(1), 49–74 (2000)
5. Dautenhahn, K., Billard, A.: Games children with autism can play with robota, a humanoid robotic doll. In: Universal access and assistive technology, pp. 179–190. Springer (2002)
6. Larregay, G., Pinna, F., Cuello, J., Morán, D.: Diseño computacional y prototipado de un iluminador uniforme utilizando diodos led. Enviado: V Congreso Argentino de Ingeniería Mecánica (CAIM) (2016)
7. Matuszek, C., Mayton, B., Aimi, R., Deisenroth, M.P., Bo, L., Chu, R., Kung, M., LeGrand, L., Smith, J.R., Fox, D.: Gambit: An autonomous chess-playing robotic system. In: Robotics and Automation (ICRA), 2011 IEEE International Conference on. pp. 4291–4297. IEEE (2011)
8. Reza, A.M.: Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. Journal of VLSI signal processing systems for signal, image and video technology 38(1), 35–44 (2004)
9. Urting, D., Berbers, Y.: Marineblue: A low-cost chess robot. In: Robotics and Applications. pp. 76–81. Citeseer (2003)