ДОДАТОК А

Апробація результатів

6 червня 2025 рік • Суми, Україна • Молодіжна наукова ліга

Шевцов Михайло Васильович, здобувач вищої освіти факультету АКТ
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Новоселов Сергій Павлович, професор кафедри КІТАР
Харківський національний університет радіоелектроніки, Україна

РОБОТИЗОВАНІ МАНІПУЛЯТОРИ ТА КОМП'ЮТЕРНИЙ ЗІР В АВТОМАТИЗАЦІЇ СОРТУВАННЯ

На сьогоднішній день одним із найбільш актуальних напрямків автоматизації виробничих процесів є робототехніка. Цей напрям полягає в інтеграції роботизованих систем у виробничі процеси для заміни або підтримки людської праці. Для належного виконання завдань в автоматичному режимі, ці системи повинні мати засоби збору та аналізу інформації про виробничий процес та навколишнє середовище. Так само як і люди, роботи можуть сприймати та «розуміти» навколишній світ завдяки технологіям комп'ютерного зору та обробки зображень. Застосування цих технологій в автоматизації виробничих процесів постійно зростає. Традиційне промислове обладнання та машини поступово замінюютьюся сучасними версіями, які поєднують роботизовані системи з автоматичним управлінням на основі комп'ютерного зору [1].

Сортування об'єктів за різними ознаками є особливо актуальним напрямком застосування подібних робототехнічних систем, оскільки операції сортування й досі часто виконуються вручну, що займає багато часу та призводить до зниження точності та надійності [1]. При цьому у світі існують тисячі виробничих підприємств, де сортування об'єктів на основі різних параметрів є життєво важливою частиною виробничого процесу. Таким чином, роботи-сортувальники, які працюють за допомогою обробки зображень, можуть значно підвищити ефективність та точність сучасного виробництва, скоротити час виконання операцій сортування, а також заощадити витрати на робочу силу. [2]

Основною вимогою, що висувається до маніпуляторів для автоматизації процесу сортування є здатність здійснювати точні, швидкі та повторювані переміщення в горизонтальній площині. Водночас, важливе значення мають такі параметри, як надійність, компактність конструкції та зручність програмування. З урахуванням вищезазначених вимог для створення автоматизованої системи сортування було обрано використання роботизованого маніпулятора з кінематикою типу SCARA (Selective Compliance Assembly Robot Arm). Завдяки своїй конструкції, SCARA-роботи широко використовуються в сортувальних операціях. Їхня здатність швидко та точно переміщувати об'єкти в горизонтальній площині робить їх доцільними для завдань, де необхідно швидко сортувати та розподіляти компоненти або продукти [3]. Так, в електронній промисловості SCARA-роботи застосовуються для складання та сортування дрібних електронних

ДОДАТОК Б

Код програми

```
class Vision(QObject):
    def __init__(self, parent=None):
        super().__init__(parent)

        self.green_king_image = cv2.imread("images/green-king.jpg")
        self.orange_king_image = cv2.imread("images/orange-king.jpg")

        self._layers = 30
        self._layer = 0
        self._boards = np.zeros((self._layers, 8, 8, 4), dtype=np.float32)

    board_ready = pyqtSignal(ndarray)
    image_ready = pyqtSignal(ndarray)

    def image2board(self, image):
        image = cv2.imdecode(np.asarray(bytearray(image), dtype=np.uint8), cv2.IMREAD_COLOR)
        if image is None: return

        board = np.zeros((8, 8, 4), dtype=np.float32)

        board_image = self.find_board(image)
        h, w, _ = board_image.shape

        gray = cv2.cvtColor(board_image, cv2.COLOR_BGR2GRAY)
        pieces = self.find_pieces(gray)

        if pieces is None:
            return board

        pieces = np.int16(np.around(pieces))
        for x, y, r in pieces[0, :]:
            x1, y1 = max(0, int(x - r)), max(0, int(y - r))
            x2, y2 = min(w, int(x + r)), min(h, int(y + r))
            piece_image = board_image[y1:y2, x1:x2]
            piece_mask = np.zeros(piece_image.shape[:2], dtype=np.uint8)
            cv2.circle(piece_mask, (r, r), r, 255, thickness=-1)
            piece_image = cv2.bitwise_and(piece_image, piece_image, mask=piece_mask)

            color = self.detect_color(piece_image)
            if color is None:
                continue

            row, col = self.detect_field(x - 30, y - 35, 675 - 265, 1000 - 595)
            king = self.detect_king(piece_image)

            piece = 0
            if color == "green":
                if not king:
                    piece = 1
                else:
                    piece = 2
            elif color == "orange":
                if not king:
                    piece = 3
                else:
                    piece = 4

            board[row][col] = piece, x - 17, y - 16, r

        if self._layer >= self._layers:
            self._layer = 0
            board[:, :, 0] = mode(self._boards[:, :, :, 0], axis=0).mode
            board[:, :, 1:] = np.mean(self._boards[:, :, :, 1:], axis=0)
            self.board_ready.emit(board)
        else:
            self._boards[self._layer, :, :] = board
            self._layer += 1

        self.image_ready.emit(np.ascontiguousarray(image[222:636, 589:1003]))
```

```
    def find_board(self, image):
        board = image[205:660, 573:1023]
        return board

    def find_pieces(self, gray):
        clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
        image = clahe.apply(gray)
        image = cv2.medianBlur(image, 7)
        return cv2.HoughCircles(image, cv2.HOUGH_GRADIENT, 1.6, 30, param1=80, param2=23,
minRadius=30, maxRadius=35)

    def detect_field(self, x, y, h, w):
        min_w, max_w = 0, w
        min_h, max_h = 0, h
        min_i, max_i = 0, 7
        min_j, max_j = 0, 7

        while min_i < max_i and min_j < max_j:
            mid_w = (min_w + max_w) / 2
            mid_h = (min_h + max_h) / 2

            mid_j = (min_j + max_j) // 2
            if x < mid_w:
                max_w = mid_w
                max_j = mid_j
            else:
                min_w = mid_w
                min_j = mid_j + 1

            mid_i = (min_i + max_i) // 2
            if y < mid_h:
                max_h = mid_h
                max_i = mid_i
            else:
                min_h = mid_h
                min_i = mid_i + 1

        return max_i, max_j

    def detect_color(self, piece_image):
        piece_image = cv2.cvtColor(piece_image, cv2.COLOR_BGR2HSV)

        hist = cv2.calcHist([piece_image], [0], None, [180], [0, 180])
        orange_hist_range = np.sum(hist[10:25])  # Orange hue range
        green_hist_range = np.sum(hist[35:85])  # Green hue range

        if orange_hist_range > green_hist_range:
            return "orange"
        elif green_hist_range > orange_hist_range:
            return "green"
        else:
            return None

    def detect_king(self, piece_image):
        sift = cv2.SIFT_create()
        # FLANN parameters
        FLANN_INDEX_KDTREE = 1
        index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
        search_params = dict(checks=50)
        flann = cv2.FlannBasedMatcher(index_params, search_params)
```

ДОДАТОК В

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮІК.412150.052 ПЗ	Пояснювальна записка	А4, 76 с.
		Додаткові матеріали	
2		Демонстраційний матеріал у	А4, 18 с.
		вигляді презентації	
		Макети, моделі	
3		Макет пристрою	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮІК.412150.052 ВД			
Розроб.		Шевцов М.В.			Розроблення роботизованого маніпулятора для автоматизації виробничої операції сортування із використанням технологій комп'ютерного зору	Літера	Аркуш	Аркушів
Перевір.		Новоселов С.П.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська Н.П.						
Затв.		Невлюдов І.Ш.						