(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년09월22일
(11) 등록번호 10-2446521
(24) 등록일자 2022년09월20일

(51) 국제특허분류(Int. Cl.)
G06K 9/00 (2022.01) *G06F 3/023* (2006.01)
G06F 40/279 (2020.01) *G06F 9/54* (2018.01)
G06K 9/62 (2022.01)
(52) CPC특허분류
G06V 30/40 (2022.01)
G06F 3/0233 (2013.01)
(21) 출원번호 10-2020-7028124
(22) 출원일자(국제) 2020년09월21일
심사청구일자 2020년10월29일
(85) 번역문제출일자 2020년09월28일
(65) 공개번호 10-2021-0134483
(43) 공개일자 2021년11월10일
(86) 국제출원번호 PCT/US2020/051878
(87) 국제공개번호 WO 2021/221713
국제공개일자 2021년11월04일
(30) 우선권주장
16/864,708 2020년05월01일 미국(US)
(56) 선행기술조사문헌
US09213625 B1*
US20190324781 A1*
US20150339213 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
유아이패스, 인크.
미국 뉴욕주 10018 뉴욕 5번가 452 22층
(72) 발명자
디네스 다니엘
미국 뉴욕주 10016 뉴욕 20번 플로어 파크 애비뉴 90
(74) 대리인
김태홍, 김진희

전체 청구항 수 : 총 19 항 심사관 : 방인환

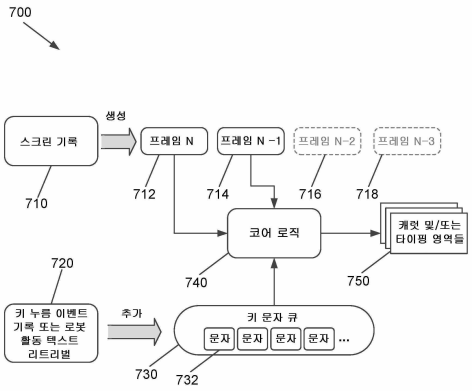
(54) 발명의 명칭 **로보틱 프로세스 자동화를 위한 로봇 실행의 스크린 응답 검증**

(57) 요 약

로봇 실행의 로보틱 프로세스 자동화(RPA)를 위한 스크린 응답 검증이 개시된다. RPA 로봇이 상호작용하는 컴퓨팅 시스템 상에서 실행되고 있는 애플리케이션에서 텍스트, 스크린 변경들, 이미지들, 및/또는 다른 예상된 시각적 액션들이 발생하는지의 여부가 인식될 수도 있다. 로봇이 타이핑한 것으로 결정될 수도 있고 하나 이상의 문

(뒷면에 계속)

대 표 도 - 도7



자, 이미지들, 윈도우들 등이 나타난 곳의 현재 해상도에 기초하여 스크린 상의 물리적 포지션이 제공될 수도 있다. 이들 엘리먼트들의 물리적 포지션, 또는 그것의 결여는, 애플리케이션과 컴퓨팅 시스템이 의도된 대로 응답하고 있다는 것의 검증 목적으로 로봇이 입력하고 있는 필드(들) 및 연관된 애플리케이션이 무엇인지의 결정을 허용할 수도 있다. 예상된 스크린 변경들이 발생하지 않는 경우, 로봇은 중지하고 예외를 발생시키며, 돌아가서 의도된 상호작용을 다시 시도하며, 워크플로우를 재시작하거나, 또는 다른 적합한 액션을 취할 수 있다.

(52) CPC특허분류
G06F 40/279 (2020.01)
G06F 9/543 (2013.01)
G06V 10/751 (2022.01)
G06V 10/759 (2022.01)
G06V 30/10 (2022.01)

명 세 서

청구범위

청구항 1

비일시적인 컴퓨터 판독가능 매체로서,

상기 비일시적인 컴퓨터 판독가능 매체는 로보틱 프로세스 자동화(robotic process automation; RPA)를 위한 스크린 응답 검증 로직을 포함하는 컴퓨터 프로그램을 저장하고, 상기 컴퓨터 프로그램은 적어도 하나의 프로세서가,

현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하는지의 여부를 결정하게 하도록 - 상기 현재 스크린샷 프레임 및 상기 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함함 - ;

상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이에 차이가 존재하고 RPA 로봇의 워크플로우(workflow)에서의 활동(activity)에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 예상될 때,

상기 RPA 로봇의 워크플로우의 활동에 기초하여 하나 이상의 예상된 스크린 변경에 대해 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 하나 이상의 변경을 검증하게 하도록, 그리고

상기 검증이 실패했을 때, 예외(exception)를 발생시키거나(throw) 또는 시정 조치(remedial action)를 개시하게 하도록

구성되고,

상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이하지 않고 상기 RPA 로봇의 워크플로우에서의 활동에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 발생할 것으로 예상되지 않을 때, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

다음 스크린샷 프레임을 페치(fetch)하게 하도록; 그리고

상기 다음 스크린샷 프레임을 상기 현재 스크린샷 프레임으로서 사용하고 상기 현재 스크린샷 프레임을 상기 이전 스크린샷 프레임으로서 사용하여 상기 프로세스를 반복하게 하도록

구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 2

삭제

청구항 3

제1항에 있어서, 상기 시정 조치는, 상기 RPA 로봇이, i) 상기 로봇이 상호작용하려고 시도하고 있는 애플리케이션용 윈도우가 액티브 윈도우(active window)인지의 여부를 드라이버 레벨(driver level)에서 체크하는 것, ii) 변경이 발생할 것으로 예상되었던 그래픽 엘리먼트에 대한 마우스 포인터 또는 캐럿(caret)의 로케이션을 체크하는 것, iii) 올바른 그래픽 엘리먼트가 액티브 엘리먼트인지의 여부를 체크하는 것, 또는 i), ii) 및 iii)의 임의의 조합을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 4

제1항에 있어서, 상기 시정 조치는, 상기 RPA 로봇이, i) 올바른 윈도우를 액티브, 포커싱된 윈도우로서 설정하는 것, ii) 포커스를 액티브 엘리먼트에 설정하는 것, iii) 마우스를 이동시키는 것, iv) 마우스 클릭 이벤트를 유발하는 것, 또는 i), ii), iii) 및 iv)의 임의의 조합을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 5

제1항에 있어서, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

시간 윈도우(time window) 동안 발생했던 상기 RPA 로봇에 의해 생성된 키 누름 이벤트(key press event)들을 포함하는 키 문자 큐(key character queue)를 생성하게 하도록 구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 6

제5항에 있어서, 상기 RPA 로봇은, 키 누름 이벤트들 사이에 시구간(time period)을 대기하도록 구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 7

제5항에 있어서, 상기 RPA 로봇의 워크플로우에서의 활동에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 발생할 것으로 예상되고 상기 RPA 로봇의 워크플로우의 활동에 기초한 상기 하나 이상의 예상된 스크린 변경이 하나 이상의 문자를 포함할 때, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이한, 하나 이상의 로케이션 상에서 광학적 문자 인식(optical character recognition; OCR)을 수행하게 하도록 구성되고,

상기 검증은, 상기 OCR에 의해 인식된 문자들을 상기 키 문자 큐에서의 문자들과 매칭시키는 것을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 8

제7항에 있어서, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 키 문자 큐로부터 매칭 문자들을 제거하게 하도록 구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 9

제5항에 있어서, 상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이하고 상기 RPA 로봇의 워크플로우에서의 활동에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 발생할 것으로 예상될 때, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이한, 하나 이상의 영역을 결정하게 하도록; 그리고

상기 결정된 하나 이상의 영역에서 하나 이상의 연결된 멤버를 추출하고 상기 추출된 하나 이상의 연결된 멤버에 대해 광학적 문자 인식(OCR)을 수행하도록 하여, i) 하나 이상의 인식된 문자, ii) 하나 이상의 인식된 문자 시퀀스, 또는 i) 및 ii)의 둘 다, 및 각각의 포지션들을 생성하도록

구성되고,

상기 검증은, 상기 i) 하나 이상의 인식된 문자, 상기 ii) 하나 이상의 인식된 문자 시퀀스, 또는 i) 및 ii)의 둘 다를, 상기 키 문자 큐 또는 상기 활동에 기초하여 나타날 것으로 예상되는 iii) 하나 이상의 문자, iv) 하나 이상의 문자 시퀀스, 또는 iii) 및 iv)의 둘 다와 비교하는 것을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 10

제1항에 있어서, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 RPA 로봇의 워크플로우에서의 활동에 기초하여 타겟 그래픽 엘리먼트 및 상기 타겟 그래픽 엘리먼트에 입력될 콘텐츠를 결정하게 하도록;

상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 하나 이상의 변경이 상기 타겟 그래픽 엘리먼트의 로케이션과 매칭되는지의 여부 및 상기 하나 이상의 변경이 상기 타겟 그래픽 엘리먼트에 입력될 콘텐츠와 매칭되는지의 여부를 검증하게 하도록; 그리고

i) 상기 로케이션, ii) 상기 하나 이상의 변경, 또는 i) 및 ii)의 둘 다가 매칭되지 않을 때, 상기 예외를 발

생시키거나 또는 상기 시정 조치를 개시하게 하도록

구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 11

제1항에 있어서, 상기 활동은, i) 그래픽 엘리먼트에 텍스트를 입력하는 것, ii) 상기 그래픽 엘리먼트로 캐럿을 이동시키는 것, iii) 상기 그래픽 엘리먼트에 콘텐츠를 붙여 넣는(paste) 것, 또는 i), ii) 및 iii)의 조합을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 12

제1항에 있어서, 상기 RPA 로봇의 워크플로우의 활동에 기초한 상기 하나 이상의 예상된 스크린 변경은, 양식(form)의 제출(submission), 새로운 애플리케이션 윈도우의 열기, 또는 현재 애플리케이션 윈도우의 변경을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 13

비일시적 컴퓨터 판독가능 매체로서,

상기 비일시적 컴퓨터 판독가능 매체는 로보틱 프로세스 자동화(RPA)를 위한 스크린 응답 검증 로직을 포함하는 컴퓨터 프로그램을 저장하고, 상기 컴퓨터 프로그램은 적어도 하나의 프로세서가,

RPA 로봇의 워크플로우에서의 활동에 기초하여 타겟 그래픽 엘리먼트 및 상기 타겟 그래픽 엘리먼트에 입력될 콘텐츠를 결정하게 하도록;

현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재하는지의 여부를 결정하게 하도록 - 상기 현재 스크린샷 프레임 및 상기 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함함 - ; 그리고

상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재하지 않을 때, 예외를 발생시키거나 또는 시정 조치를 개시하게 하도록

구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 14

제13항에 있어서, 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재할 때, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 변경이 상기 타겟 그래픽 엘리먼트의 로케이션과 매칭되는지의 여부 및 상기 변경이 상기 타겟 그래픽 엘리먼트에 입력될 콘텐츠와 매칭되는지의 여부를 검증하게 하도록; 그리고

i) 상기 로케이션, ii) 상기 변경, 또는 i) 및 ii)의 둘 다가 매칭되지 않을 때, 예외를 발생시키거나 또는 시정 조치를 개시하게 하도록

구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 15

제13항에 있어서, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 사용자 인터페이스에서 그래픽 엘리먼트들 및 연관된 경계 직사각형(bounding rectangle)들을 결정하도록 컴퓨터 비전 알고리즘을 적용하게 하도록 구성되고, 상기 결정된 그래픽 엘리먼트들은 상기 타겟 그래픽 엘리먼트를 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 16

제13항에 있어서, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

시간 윈도우 동안 발생했던 상기 RPA 로봇에 의해 생성된 키 누름 이벤트들을 포함하는 키 문자 큐를 생성하게

하도록 구성되는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 17

제16항에 있어서, 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재할 때, 상기 컴퓨터 프로그램은 또한, 상기 적어도 하나의 프로세서가,

상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이한, 상기 하나 이상의 로케이션 상에서 광학적 문자 인식(OCR)을 수행하게 하도록 구성되고,

상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 변경이 상기 타겟 그래픽 엘리먼트에 입력될 콘텐츠와 매칭되는지의 여부의 검증은, 상기 OCR에 의해 인식된 문자들을 상기 키 문자 큐에서의 문자들과 매칭시키는 것을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 18

제13항에 있어서, 상기 RPA 로봇의 워크플로우의 활동에 기초한 상기 하나 이상의 예상된 스크린 변경은, 양식의 제출, 새로운 애플리케이션 윈도우의 열기, 또는 현재 애플리케이션 윈도우의 변경을 포함하는 것인, 비일시적 컴퓨터 판독가능 매체.

청구항 19

컴퓨터로 구현되는 방법(computer-implemented method)으로서,

상기 컴퓨터로 구현되는 방법은, 로보틱 프로세스 자동화(RPA)를 위한 스크린 응답 검증을 수행하기 위한 것이고,

RPA 로봇에 의해, 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하는지의 여부를 결정하는 단계 - 상기 현재 스크린샷 프레임 및 상기 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함함 - ; 및

상기 현재 스크린샷 프레임이 상기 이전 스크린샷 프레임과 상이하지 않고 상기 RPA 로봇의 워크플로우에서의 활동에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 발생할 것으로 예상될 때, 상기 RPA 로봇에 의해, 예외를 발생시키거나 또는 시정 조치를 개시하는 단계

를 포함하는, 컴퓨터로 구현되는 방법.

청구항 20

제19항에 있어서, 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이에 차이가 존재하고 RPA 로봇의 워크플로우에서의 활동에 기초하여 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 차이가 예상될 때,

상기 RPA 로봇에 의해, 상기 RPA 로봇의 워크플로우의 활동에 기초하여 하나 이상의 예상된 스크린 변경에 대해 상기 현재 스크린샷 프레임과 상기 이전 스크린샷 프레임 사이의 하나 이상의 변경을 검증하는 단계; 및

상기 검증이 실패했을 때, 상기 RPA 로봇에 의해 예외를 발생시키거나 또는 시정 조치를 개시하는 단계

를 더 포함하는, 컴퓨터로 구현되는 방법.

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

발명의 설명

기 술 분 야

[0001] 본 출원은 2020년 5월 1일자로 출원된 미국 정식 특허출원 제16/864,708호를 우선권 주장한다. 이전에 출원된 이 출원의 발명의 주제는 이로써 그 전부가 참조로 본원에 포함된다.

[0002] 본 발명은 대체로 로보틱 프로세스 자동화(robotic process automation; RPA)에 관한 것이고, 더 구체적으로는, RPA를 위한 로봇 실행의 스크린 응답 검증에 관한 것이다.

배 경 기 술

[0003] RPA 로봇들은 사용자들과 유사한 방식으로 컴퓨팅 시스템들과 상호작용할 수도 있다. 예를 들면, 로봇들은 마우스를 움직이며, 텍스트를 입력하며, 버튼들을 클릭하는 등을 할 수도 있다. 그러나, 인간 사용자들과는 달리, RPA 로봇들은 그들의 상호작용들에 응답하여 컴퓨팅 시스템이 하고 있는 것을 "알" 수 없고, 심지어 컴퓨팅 시스템이 의도된 대로 응답하고 있지 않더라도 RPA 로봇들은 그들의 로직에 따라 진행할 수도 있다. 따라서, 컴퓨팅 시스템과의 RPA 로봇 상호작용들을 모니터링하고 검증하는 개선된 접근법이 유익할 수도 있다.

발명의 내용

[0004] 본 발명의 특정한 실시예들은 현재 RPA 기술들에 의해 아직 완전히 식별, 인식, 또는 해결되지 않은 업계에서의 문제들 및 요구들에 대한 해결책들을 제공할 수도 있다. 예를 들어, 본 발명의 일부 실시예들은 RPA를 위한 로봇 실행의 스크린 응답 검증(validation)에 관한 것이다.

[0005] 일 실시예에서, RPA를 위한 스크린 응답 검증 로직을 포함하는 컴퓨터 프로그램이 비일시적 컴퓨터 판독가능 매체 상에 구현된다. 그 프로그램은 적어도 하나의 프로세서로 하여금 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하는지의 여부를 결정하게 하도록 구성된다. 현재 스크린샷 프레임과 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함한다. 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하고 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이의 차이가 RPA 로봇의 워크플로우(workflow)에서의 활동(activity)에 기초하여 예상될 때, 프로그램은 또한 적어도 하나의 프로세서로 하여금, RPA 로봇의 워크플로우의 활동에 기초하여 하나 이상의 예상된 스크린 변경에 대해 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이의 하나 이상의 변경을 검증하고, 검증이 실패했을 때, 예외(exception)를 발생시키거나(throw) 또는 시정 조치를 개시하게 하도록 구성된다.

[0006] 다른 실시예에서, RPA를 위한 스크린 응답 검증 로직을 포함하는 컴퓨터 프로그램이 비일시적 컴퓨터 판독가능 매체 상에 구현된다. 그 프로그램은 적어도 하나의 프로세서로 하여금, RPA 로봇의 워크플로우에서의 활동에 기초하여 타겟 그래픽 엘리먼트 및 타겟 그래픽 엘리먼트에 입력될 콘텐츠를 결정하고, 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재하는지의 여부를 결정하게 하도록 구성된다. 현재 스크린샷 프레임과 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함한다. 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 하나 이상의 차이가 존재하지 않을 때, 그 프로그램은 또한 적어도 하나의 프로세서로 하여금 예외를 발생시키거나 또는 시정 조치를 개시하게 하도록 구성된다.

[0007] 또 다른 실시예에서, RPA를 위한 스크린 응답 검증을 수행하기 위한 컴퓨터로 구현되는 방법(computer-implemented method)이, RPA 로봇에 의해, 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하는지의 여부를 결정하는 단계를 포함한다. 현재 스크린샷 프레임과 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함한다. 현재 스크린샷 프레임이 이전 스크린샷 프레임과 상이하지 않고 RPA 로봇의 워크플로우에서의 활동에 기초하여 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이의 차이가 발생할 것으로 예상될 때, 컴퓨터로 구현되는 방법은, RPA 로봇에 의해, 예외를 발생시키거나 또는 시정 조치를 개시하는 단계를 포함한다.

도면의 간단한 설명

[0008] 본 발명의 특정한 실시예들의 장점들이 쉽사리 이해되도록 하기 위하여, 위에서 간략히 설명된 본 발명의 더 특

정한 설명이 첨부된 도면들에서 예시되는 특정 실시예들을 참조하여 제공될 것이다. 이들 도면들은 본 발명의 전형적인 실시예들만을 묘사하고 그러므로 본 발명의 범위의 제한으로 간주되지 않는 것으로 이해되어야 하지만, 본 발명은 첨부 도면들의 사용을 통해 추가의 특이성 및 세부사항과 함께 기재되고 설명될 것이며, 도면들 중:

도 1은 본 발명의 일 실시예에 따른 로보틱 프로세스 자동화(RPA) 시스템을 예시하는 아키텍처 도면이다.

도 2는 본 발명의 일 실시예에 따른 전개된(deployed) RPA 시스템(200)을 예시하는 아키텍처 도면이다.

도 3은 본 발명의 일 실시예에 따른 설계자, 활동들, 및 드라이버들 사이의 관계를 예시하는 아키텍처 도면이다.

도 4는 본 발명의 일 실시예에 따른 RPA 시스템을 예시하는 아키텍처 도면이다.

도 5는 본 발명의 일 실시예에 따른, RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하도록 구성되는 컴퓨팅 시스템(500)을 예시하는 아키텍처 도면이다.

도 6a 및 도 6b는 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 위한 프로세스를 예시하는 흐름도들이다.

도 7은 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 위한 프로세스를 예시하는 아키텍처 도면이다.

도 8a는 본 발명의 일 실시예에 따른, 각각 프레임 N - 1 및 프레임 N에 대한 동문 기부 윈도우를 포함하는 스크린샷들의 부분들을 예시한다.

도 8b는 본 발명의 일 실시예에 따른 정사각형 그리드로 분할된 스크린샷 N - 1을 예시한다.

도 8c는 본 발명의 일 실시예에 따른 정사각형 그리드로 분할된 스크린샷 N을 예시한다.

도 8d는 본 발명의 일 실시예에 따른 스크린샷 N - 1로부터의 변경을 포함하는 강조표시(highlight)된 행을 갖는 스크린샷 N을 예시한다.

도 8e는 본 발명의 일 실시예에 따른 스크린샷 N - 1로부터의 변경을 포함하는 강조표시된 행에서 강조표시된 셀을 갖는 스크린샷 N을 예시한다.

도 8f는 본 발명의 일 실시예에 따른 스크린샷 N - 1로부터의 변경을 포함하는 강조표시된 영역을 갖는 스크린샷 N을 예시한다.

도 9는 본 발명의 일 실시예에 따른 해상도 변경들에 대한 체크, 연결된 디스플레이들의 범위에서의 변경들에 대한 체크, 및 변경들을 수용하기 위한 캐럿 추적 비디오 로직(caret tracking video logic) 설정을 위한 프로세스를 예시하는 흐름도이다.

도 10은 본 발명의 일 실시예에 따른 비디오 레코더를 위한 프로세스를 예시하는 흐름도이다.

도 11은 본 발명의 일 실시예에 따른 캐럿 추적을 수행하기 위한 프로세스를 예시하는 흐름도이다.

도 12a 내지 도 12d는 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하기 위한 프로세스를 예시하는 흐름도들이다.

도 13a 내지 도 13c는 본 발명의 일 실시예에 따른 붙여 넣은(paste) 텍스트 추적 및 검증을 수행하기 위한 프로세스를 예시하는 흐름도들이다.

도 14a는 본 발명의 일 실시예에 따른 32×32 영역으로 결합된 네 개의 16×16 블록들에 포함되는 글자 "E"에 대한 예시적인 이진 매트릭스를 도시한다.

도 14b는 본 발명의 일 실시예에 따른 도 14a의 영역을 위한 이진 매트릭스에 포함되는 멤버(글자 "E")에 대한 크로핑된 7×9 이진 매트릭스를 예시한다.

도 15는 본 발명의 일 실시예에 따른 CV와 키 큐를 사용하여 액티브 엘리먼트들을 결정하고 로봇 액션들을 검증하기 위한 프로세스를 예시하는 흐름도이다.

도 16은 본 발명의 일 실시예에 따른, RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하도록 구성되는 시스템

(500)을 예시하는 아키텍처 도면이다.

도 17은 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하기 위한 프로세스를 예시하는 흐름도이다.

달리 지시되지 않는 한, 유사한 참조 부호들은 결부된 도면들의 전체에 걸쳐 일관되게 대응하는 특징들을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0009] 일부 실시예들은 RPA를 위한 로봇 실행의 스크린 응답 검증에 관한 것이다. 일부 실시예들은 텍스트, 스크린 변경들, 이미지들, 및/또는 다른 예상된 시각적 액션들이 RPA 로봇이 상호작용하는 컴퓨팅 시스템 상에서 실행되고 있는 애플리케이션에서 발생하는지의 여부를 인식한다. 이러한 실시예들은 컴퓨팅 시스템에 연관되는 스크린 상에서, 로봇이 입력한 곳을 찾고 하나 이상의 문자, 이미지, 윈도우 등이 나타난 곳의 (예컨대, 좌표들에서의) 현재 해상도에 기초하여 스크린 상의 물리적 포지션을 제공할 수도 있다. 이들 엘리먼트들의 물리적 포지션은, 애플리케이션과 컴퓨팅 시스템이 의도된 대로 응답하고 있다는 것의 검증 목적으로 로봇이 입력하고 있는 필드(들) 및 연관된 애플리케이션이 무엇인지의 결정을 허용할 수도 있다. 나타나는 문자들, 이미지들, 윈도우들 등의 부족과, 또는 이들 엘리먼트들이 잘못된 장소에서 나타나거나 또는 잘못된 유형을 가진다는 것은 이 분석에서 결정될 수도 있고, 로봇은 정지하고 예외를 발생시키며, 돌아가서 의도된 상호작용을 다시 시도하며, 워크플로우를 재시작하거나, 또는 컴퓨팅 시스템 상의 애플리케이션이 워크플로우의 기능을 구현하고 있지 않으면 임의의 다른 적합한 액션을 발명의 범위로부터 벗어나는 일 없이 취할 수 있다.

[0010] 일부 실시예들은 현재 스크린샷과 이전 스크린샷을 연속적으로 또는 주기적으로 비교하거나, 또는 현재 스크린샷의 부분과 이전 스크린샷의 부분(예컨대, 가시적 애플리케이션 윈도우에 연관된 스크린샷의 부분)을 비교하여, 변경이 발생하였는지의 여부와, 그렇다면, 그 변경들이 무엇이었는지를 확인하는 피드백 루프 프로세스로 구현된다. 윈도우들에 연관된 스크린샷들의 부분들을 비교하는 특정 실시예들에서, 로봇이 사용자와 함께 작업하는 유인 로봇일 때, 알고리즘은 윈도우의 로케이션, 윈도우의 사이즈, 윈도우의 모양, 윈도우에 대한 줌 변수, 이들의 조합 등에 대한 변경들을 수용할 수도 있어서, 그 알고리즘은 스크린샷의 현재 부분이 스크린샷의 이전의 부분에 매핑되는 것으로 인식한다.

[0011] 입력된 텍스트 검증의 경우, 스크린 상에서 시각적 변경들이 발생했던 로케이션(들)은 식별될 수도 있고 광학적 문자 인식(optical character recognition; OCR)은 변경이 발생했던 로케이션(들)에 대해 수행될 수도 있다. 그 다음에 OCR의 결과들은 매칭이 존재하는지의 여부를 결정하기 위해 (예컨대, RPA 로봇에 의해 발생된 키 누름 이벤트(key press event)들에 의해 결정된 바와 같이) 큐의 콘텐츠와 비교될 수도 있다. 변경이 발생하였던 로케이션들은 현재 스크린샷으로부터의 화소 박스와 이전 스크린샷으로부터의 동일한 로케이션에서의 화소 박스를 비교함으로써 결정될 수도 있다.

[0012] 특정한 실시예들은 로보틱 프로세스 자동화(RPA)를 위해 채용될 수도 있다. 도 1은 본 발명의 일 실시예에 따른 RPA 시스템(100)을 예시하는 아키텍처 도면이다. RPA 시스템(100)은 개발자가 워크플로우들을 설계하고 구현하는 것을 허용하는 설계자(110)를 포함한다. 설계자(110)는 애플리케이션 통합, 뿐만 아니라 서드파티 애플리케이션들, 관리 정보 기술(IT) 태스크들, 및 비즈니스 IT 프로세스들의 자동화를 위한 솔루션을 제공할 수도 있다. 설계자(110)는 비즈니스 프로세스의 그래픽 표현인 자동화 프로젝트의 개발을 용이하게 할 수도 있다. 간단히 말해서, 설계자(110)는 워크플로우들 및 로봇들의 개발 및 전개를 용이하게 한다.

[0013] 자동화 프로젝트는 본 개시에서 "활동들(activities)"로서 정의되는 워크플로우로 개발된 사용자지정 세트의 단계들 사이의 실행 순서 및 관계의 제어를 개발자에게 제공함으로써 규칙 기반 프로세스들의 자동화를 가능하게 한다. 설계자(110)의 일 실시예의 하나의 상업적 예가 UiPath Studio™이다. 각각의 활동은 버튼 클릭, 파일 판독, 로그 패널에의 기입 등과 같은 액션을 포함할 수도 있다. 일부 실시예들에서, 워크플로우들은 네스트되거나 또는 삽입될 수도 있다.

[0014] 일부 유형들의 워크플로우들은 시퀀스들, 흐름도들, FSM들, 및/또는 전역 예외 핸들러들을 비제한적으로 포함할 수도 있다. 시퀀스들은 워크플로우를 혼잡하게 하는 일 없이 하나의 활동에서부터 다른 활동으로의 흐름을 가능하게 하는 선형 프로세스들에 특히 적합할 수도 있다. 흐름도들은 더 복잡한 비즈니스 로직에 특히 적합하여, 다수의 분기 로직 오퍼레이터들을 통해 더 다양한 방식으로 활동들의 연결과 결정들의 통합을 가능하게 할 수도 있다. FSM들은 큰 워크플로우들에 특히 적합할 수도 있다. FSM들은 조건(즉, 전이(transition)) 또는 활동에 의해 트리거되는 유한한 수의 상태들을 자신들의 실행에 사용할 수도 있다. 전역 예외 핸들러들은

실행 에러와 마주칠 때에 워크플로우 거동을 결정하는데 그리고 프로세스들을 디버깅하는데 특히 적합할 수도 있다

[0015] 일단 워크플로우가 설계자(110)에서 개발되면, 비즈니스 프로세스들의 실행은 수행자(conductor)(120)에 의해 오케스트레이션되는데, 수행자는 설계자(110)에서 개발된 워크플로우들을 실행하는 하나 이상의 로봇(130)을 오케스트레이션한다. 수행자(120)의 일 실시예의 하나의 상업적 예는 UiPath Orchestrator™이다. 수행자(120)는 환경에서 리소스들의 생성, 모니터링 및 전개의 관리를 용이하게 한다. 수행자(120)는 서드파티 솔루션들 및 애플리케이션들을 이용하여, 통합 지점, 또는 집성 지점들 중 하나의 집성 지점으로서 역할을 할 수도 있다.

[0016] 수행자(120)는 로봇들(130)의 플릿(fleet)을 관리하여, 중앙 지점으로부터의 로봇들(130)을 연결하고 실행할 수도 있다. 관리될 수도 있는 로봇들(130)의 유형들은 유인 로봇들(132), 무인 로봇들(134), 개발 로봇들(무인 로봇들(134)과 유사하지만, 개발 및 테스트 목적들로 사용됨), 및 비생산 로봇들(유인 로봇들(132)과 유사하지만, 개발 및 테스트 목적들로 사용됨)을 비제한적으로 포함한다. 유인 로봇들(132)은 사용자 이벤트들에 의해 트리거되고 동일한 컴퓨팅 시스템 상의 사람과 함께 동작한다. 유인 로봇들(132)은 중앙집중형 프로세스 전개 및 로깅 매체를 위해 수행자(120)와 함께 사용될 수도 있다. 유인 로봇들(132)은 인간 사용자가 다양한 태스크들을 수행하도록 도울 수도 있고, 사용자 이벤트들에 의해 트리거될 수도 있다. 일부 실시예들에서, 프로세스들은 이 유형의 로봇에 대한 수행자(120)로부터 시작될 수 없으며 그리고/또는 그들 프로세스들은 잠긴 스크린 하에서 실행될 수 없다. 특정 실시예들에서, 유인 로봇들(132)은 오직 로봇 트레이 또는 커맨드 프롬프트로부터만 시작될 수 있다. 유인 로봇들(132)은 일부 실시예들에서 인간의 감독 하에 실행되어야 한다.

[0017] 무인 로봇들(134)은 가상 환경들에서 무인으로 실행되고 많은 프로세스들을 자동화할 수 있다 무인 로봇들(134)은 원격 실행, 모니터링, 스케줄링, 및 작업 큐들에 대한 지원의 제공을 담당할 수도 있다. 모든 로봇 유형들에 대한 디버깅은 일부 실시예들에서 설계자(110)에서 실행될 수 있다. 유인 로봇 및 무인 로봇 둘 다는 메인프레임들, 웹 애플리케이션들, VM들, 엔터프라이즈 애플리케이션들(예컨대, SAP®, SalesForce®, Oracle® 등에 의해 생성된 것들), 및 컴퓨팅 시스템 애플리케이션들(예컨대, 데스크톱 및 랩톱 애플리케이션들, 모바일 디바이스 애플리케이션들, 착용가능 컴퓨터 애플리케이션들 등)을 비제한적으로 포함하는 다양한 시스템들 및 애플리케이션들을 자동화할 수도 있다.

[0018] 수행자(120)는 프로비저닝(provisioning), 전개, 버저닝(versioning), 구성, 큐잉, 모니터링, 로깅, 및/또는 상호연결성의 제공을 비제한적으로 포함하는 다양한 능력들을 가질 수도 있다. 프로비저닝은 로봇들(130)과 수행자(120)(예컨대, 웹 애플리케이션) 사이의 연결들의 생성 및 유지보수를 포함할 수도 있다. 전개는 패키지 버전들의 배정된(assigned) 로봇들(130)로의 올바른 전달을 실행을 위해 보장하는 것을 포함할 수도 있다. 버저닝은 일부 실시예들에서 일부 프로세스 또는 구성의 고유 인스턴스들의 관리를 포함할 수도 있다. 구성은 로봇 환경들 및 프로세스 구성들의 유지보수 및 전달을 포함할 수도 있다. 큐잉은 큐들 및 큐 아이템들의 관리를 제공하는 것을 포함할 수도 있다. 모니터링은 로봇 식별 데이터의 추적을 유지하는 것과 사용자 허가들을 유지하는 것을 포함할 수도 있다. 로깅은 데이터베이스(예컨대, SQL 데이터베이스) 및/또는 다른 저장 메커니즘(예컨대, 큰 데이터세트들을 저장하고 빠르게 쿼리 할 수 있는 기능을 제공하는 ElasticSearch®)에 대한 로그들의 저장 및 인덱싱을 포함할 수도 있다. 수행자(120)는 서드파티 솔루션들 및/또는 애플리케이션들에 대한 중앙 통신 지점으로서 역할을 함으로써 상호연결성을 제공할 수도 있다.

[0019] 로봇들(130)은 설계자(110)에 내장된 워크플로우들을 실행하는 실행 에이전트들이다. 로봇(들)(130)의 일부 실시예들의 하나의 상업적 예는 UiPath Robots™이다. 일부 실시예들에서, 로봇들(130)은 Microsoft Windows® 서비스 제어 관리자(Service Control Manager; SCM) 관리 서비스를 디폴트로 설치한다. 그 결과, 이러한 로봇들(130)은 국부 시스템 계정 하에 상호작용 Windows® 세션들을 열 수 있고, Windows® 서비스의 권한을 가진다.

[0020] 일부 실시예들에서, 로봇들(130)은 사용자 모드로 설치될 수 있다. 이러한 로봇들(130)의 경우, 이는 그들 로봇들이 주어진 로봇(130)이 설치한 사용자와 동일한 권한을 가짐을 의미한다. 이 특징은 또한 각각의 머신의 완전한 이용을 자신의 최대 잠재력에서 보장하는 고밀도(High Density; HD) 로봇들에 이용 가능할 수도 있다. 일부 실시예들에서, 임의의 유형의 로봇(130)은 HD 환경에서 구성될 수도 있다.

[0021] 일부 실시예들에서의 로봇들(130)은 각각의 컴포넌트가 특정 자동화 태스크에 전용되는 여러 컴포넌트들로 분할된다. 일부 실시예들에서의 로봇 컴포넌트들은 SCM 관리 로봇 서비스들, 사용자 모드 로봇 서비스들, 실행자들, 에이전트들, 및 커맨드 라인을 비제한적으로 포함한다. SCM 관리 로봇 서비스들은 Windows® 세션들

을 관리하고 모니터링하며 수행자(120)와 실행 호스트들(즉, 로봇들(130)이 실행되는 컴퓨팅 시스템들) 사이의 프록시로서 역할을 한다. 이들 서비스들은 로봇들(130)에 대한 자격증명을 신뢰하고 관리한다. 콘솔 애플리케이션이 국부 시스템 하에서 SCM에 의해 시작된다.

[0022] 일부 실시예들에서의 사용자 모드 로봇 서비스들은 Windows® 세션들을 모니터링하고 관리하며 수행자(120)와 실행 호스트들 사이의 프록시로서 역할을 한다. 사용자 모드 로봇 서비스들은 로봇들(130)에 대한 자격증명을 신뢰하고 관리한다. SCM 관리 로봇 서비스가 설치되지 않으면 Windows® 애플리케이션이 자동으로 시작될 수도 있다.

[0023] 실행자들은 Windows® 세션 하에서 주어진 작업들을 실행할 수도 있다(즉, 이들은 워크플로우들을 실행할 수도 있다). 실행자들은 모니터별 DPI(dots per inch) 설정들을 알고 있을 수도 있다. 에이전트들은 시스템 트레이 윈도우에 가용 작업들을 디스플레이하는 Windows® 프레젠테이션 파운데이션(WPF) 애플리케이션들일 수도 있다. 에이전트들은 서비스의 클라이언트일 수도 있다. 에이전트들은 작업들을 시작하거나 또는 중지하는 것과 설정들을 변경하는 것을 요청할 수도 있다. 커맨드 라인은 서비스의 클라이언트이다. 커맨드 라인은 작업들을 시작할 것을 요청할 수 있고 그 작업들의 출력을 기다리는 콘솔 애플리케이션이다.

[0024] 위에서 설명된 바와 같이 로봇들(130)의 컴포넌트들을 분리하면 개발자들, 지원 사용자들, 및 컴퓨팅 시스템들은 각각의 컴포넌트가 실행하고 있는 것을 더 쉽게 실행하며, 식별하고, 추적할 수 있게 한다. 실행자 및 서비스에 대해 상이한 방화벽 규칙들을 설정하는 것과 같이, 특수한 거동들은 이러한 방식으로 컴포넌트별로 구성될 수도 있다. 실행자는 일부 실시예들에서 모니터링별 DPI 설정들을 항상 알고 있을 수도 있다. 그 결과, 워크플로우들은 그들 워크플로우들이 생성되었던 컴퓨팅 시스템의 구성에 상관없이 임의의 DPI에서 실행될 수 있다. 설계자(110)로부터의 프로젝트들은 또한 일부 실시예들에서 브라우저 줌 레벨과는 독립적일 수도 있다. DPI 비인식 또는 의도적으로 비인식으로 마킹된 애플리케이션들의 경우, DPI는 일부 실시예들에서 디스에이블될 수도 있다.

[0025] 도 2는 본 발명의 일 실시예에 따른 전개된 RPA 시스템(200)을 예시하는 아키텍처 도면이다. 일부 실시예들에서, RPA 시스템(200)은 도 1의 RPA 시스템(100)일 수도 있거나, 또는 그 RPA 시스템의 일부일 수도 있다. 클라이언트 측, 서버 측, 또는 이들 둘 다는 발명의 범위로부터 벗어나는 일 없이 임의의 원하는 수의 컴퓨팅 시스템들을 포함할 수도 있다는 것에 주의해야 한다. 클라이언트 측에서, 로봇 애플리케이션(210)이 실행자들(212), 에이전트(214), 및 설계자(216)를 포함한다. 그러나, 일부 실시예들에서, 설계자(216)는 컴퓨팅 시스템(210) 상에서 실행되지 않을 수도 있다. 실행자들(212)은 프로세스들을 실행하고 있다. 여러 비즈니스 프로젝트들은, 도 2에서 도시된 바와 같이, 동시에 실행될 수도 있다. 에이전트(214)(예컨대, Windows® 서비스)가 이 실시예에서의 모든 실행자들(212)에 대한 단일 접촉 지점이다. 이 실시예에서의 모든 메시지들은 수행자(230)에 로깅되며, 수행자는 그들 메시지들을 데이터베이스 서버(240), 인덱서 서버(250), 또는 둘 다를 통해 추가로 프로세싱한다. 도 1을 참조하여 위에서 논의된 바와 같이, 실행자들(212)은 로봇 컴포넌트들일 수도 있다.

[0026] 일부 실시예들에서, 로봇이 머신 이름과 사용자 이름 사이의 연관을 표현한다. 그 로봇은 동시에 다수의 실행자들을 관리할 수도 있다. 동시에 실행하고 있는 다수의 상호작용 세션들을 지원하는 컴퓨팅 시스템들(예컨대, Windows® 서버 2012) 상에서, 다수의 로봇들은 고유 사용자이름을 사용하는 별도의 Windows® 세션에서 각각을 동일한 시간에 실행되고 있을 수도 있다. 이는 위에서의 HD 로봇들이라고 지칭된다.

[0027] 에이전트(214)는 (예컨대, 로봇이 여전히 기능하고 있음을 지시하는 "심박" 메시지를 주기적으로 전송하는) 로봇의 스테이터스를 전송하고 실행될 패키지의 요구된 버전을 다운로드하는 역할을 또한 담당한다. 에이전트(214)와 수행자(230) 사이의 통신이 일부 실시예들에서 에이전트(214)에 의해 항상 개시된다. 통지 시나리오에서, 에이전트(214)는 수행자(230)에 의해 나중에 사용되는 WebSocket 채널을 로봇에 커맨드들(예컨대, 시작, 정지 등)을 전송하기 위해 열 수도 있다.

[0028] 서버 측에는, 프레젠테이션 계층(웹 애플리케이션(232), 오픈 데이터 프로토콜(Open Data Protocol; OData) REST(Representative State Transfer) 애플리케이션 프로그래밍 인터페이스(Application Programming Interface; API) 엔드포인트들(234), 그리고 통지 및 모니터링(236)), 서비스 계층(API 구현예/비즈니스 로직(238)), 및 영속 계층(데이터베이스 서버(240) 및 인덱서 서버(250))가 포함된다. 수행자(230)는 웹 애플리케이션(232), OData REST API 엔드포인트들(234), 통지 및 모니터링(236), 그리고 API 구현예/비즈니스 로직(238)을 포함한다. 일부 실시예들에서, 사용자가 수행자(230)의 인터페이스에서 (예컨대, 브라우저(220)를 통

해) 수행하는 대부분의 액션들은 다양한 API들을 호출함으로써 수행된다. 이러한 액션들은 발명의 범위를 벗어나는 일 없이 로봇들 상의 작업들을 시작하는 것, 큐들에서 데이터를 추가/제거하는 것, 무인 실행을 위한 작업들을 스케줄링하는 것 등을 비제한적으로 포함할 수도 있다. 웹 애플리케이션(232)은 서버 플랫폼의 시각적 계층이다. 이 실시예에서, 웹 애플리케이션(232)은 하이퍼 텍스트 마크업 언어(Hypertext Markup Language; HTML)와 자바스크립트(JavaScript; JS)를 사용한다. 그러나, 임의의 원하는 마크업 언어들, 스크립트 언어들, 또는 임의의 다른 포맷들이 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다. 사용자는 다양한 액션들을 수행하여 수행자(230)를 제어하기 위하여 이 실시예에서 브라우저(220)를 통해 웹 애플리케이션(232)으로부터의 웹 페이지들과 상호작용한다. 예를 들면, 사용자는 로봇 그룹들을 생성하며, 패키지들을 로봇들에 배정하며, 로봇별 및/또는 프로세스별 로그들을 분석하며, 로봇들을 시작하고 정지하는 등을 할 수도 있다.

[0029] 웹 애플리케이션(232) 외에도, 수행자(230)는 OData REST API 엔드포인트들(234)을 노출시키는 서비스 계층을 또한 포함한다. 그러나, 다른 엔드포인트들은 발명의 범위로부터 벗어나는 일 없이 포함될 수도 있다. REST API는 웹 애플리케이션(232) 및 에이전트(214) 둘 다에 의해 소비된다. 에이전트(214)는 이 실시예에서 클라이언트 컴퓨터 상의 하나 이상의 로봇의 감독자이다.

[0030] 이 실시예에서의 REST API는 구성, 로깅, 모니터링, 및 큐잉 기능을 커버한다. 구성 엔드포인트들은 일부 실시예들에서 애플리케이션 사용자들, 허가들, 로봇들, 자산들, 릴리스들, 및 환경들을 정의하고 구성하는데 사용될 수도 있다. 로깅 REST 엔드포인트들은, 예를 들면, 에러들, 로봇들에 의해 전송된 명시적 메시지들, 및 다른 환경 특정 정보와 같은 상이한 정보를 로깅하는데 사용될 수도 있다. 전개 REST 엔드포인트들은 시작 작업 커맨드가 수행자(230)에서 사용되면 실행되어야 하는 패키지 버전을 쿼리하기 위해 로봇들에 의해 사용될 수도 있다. 큐잉 REST 엔드포인트들은 큐들과 데이터를 큐에 추가하는 것, 큐로부터 트랜잭션을 획득하는 것, 트랜잭션의 스테이터스를 설정하는 것 등과 같은 큐 아이템 관리를 담당할 수도 있다.

[0031] REST 엔드포인트들을 모니터링하면 웹 애플리케이션(232) 및 에이전트(214)를 모니터링할 수도 있다. 통지 및 모니터링 API(236)는 에이전트(214)를 등록하는데, 구성 설정들을 에이전트(214)에 전달하는데, 그리고 서버 및 에이전트(214)로부터의 통지들을 전송/수신하는데 사용되는 REST 엔드포인트들일 수도 있다. 통지 및 모니터링 API(236)는 일부 실시예들에서 WebSocket 통신을 또한 사용할 수도 있다.

[0032] 영속 계층(persistence layer)은 이 실시예에서 한 쌍의 서버들 - 데이터베이스 서버(240)(예컨대, SQL 서버) 및 인덱서 서버(250) - 을 포함한다. 이 실시예에서의 데이터베이스 서버(240)는 로봇들, 로봇 그룹들, 연관된 프로세스들, 사용자들, 규칙들, 스케줄들 등의 구성을 저장한다. 이 정보는 일부 실시예들에서 웹 애플리케이션(232)을 통해 관리된다. 데이터베이스 서버(240)는 큐들 및 큐 아이템들을 관리할 수도 있다. 일부 실시예들에서, 데이터베이스 서버(240)는 (인덱서 서버(250)에 더하여 또는 그 인덱서 서버 대신에) 로봇들에 의해 로깅된 메시지들을 저장할 수도 있다.

[0033] 인덱서 서버(250)는, 일부 실시예들에서 옵션적인 것으로, 로봇들에 의해 로깅된 정보를 저장하고 인덱싱한다. 특정 실시예들에서, 인덱서 서버(250)는 구성 설정들을 통해 디스에이블될 수도 있다. 일부 실시예들에서, 인덱서 서버(250)는 ElasticSearch®를 사용하는데, 이는 오픈 소스 프로젝트 풀 텍스트 서치 엔진(open source project full-text search engine)이다. 로봇들(예컨대, 로그 메시지 또는 기입 라인과 같은 활동들을 사용함)에 의해 로깅된 메시지들은 로깅 REST 엔드포인트(들)를 통해 인덱서 서버(250)에 전송될 수도 있으며, 인덱서 서버에서 그 메시지들은 추가의 이용을 위해 인덱싱된다.

[0034] 도 3은 본 발명의 일 실시예에 따른 설계자(310), 활동들(320, 330), 및 드라이버들(340) 사이의 관계(300)를 예시하는 아키텍처 도면이다. 위와 같이, 개발자가 설계자(310)를 사용하여 로봇들에 의해 실행되는 워크플로우들을 개발한다. 워크플로우들은 사용자 정의 활동들(320)과 UI 자동화 활동들(330)을 포함할 수도 있다. 일부 실시예들은 본 개시에서 컴퓨터 비전(computer vision; CV)이라 불리는, 이미지에서의 비 텍스트 시각적 컴포넌트들을 식별할 수 있다. 이러한 컴포넌트들에 관련된 일부 CV 활동들은 클릭, 타이핑(type), 텍스트 가져오기, 호버(hover), 엘리먼트 존재, 범위 리프레시, 강조표시 등을 비제한적으로 포함할 수도 있다. 일부 실시예들에서의 클릭은, 예를 들어, CV, 광학적 문자 인식(OCR), 퍼지 텍스트 매칭, 및 멀티 앵커를 사용하여 엘리먼트를 식별하고, 그 엘리먼트를 클릭한다. 타이핑은 위의 것을 사용하여 엘리먼트를 식별하고 그 엘리먼트에 입력할 수도 있다. 텍스트 가져오기는 특정 텍스트의 로케이션을 식별하고 그 특정 텍스트를 OCR을 사용하여 스캔할 수도 있다. 호버는 엘리먼트를 식별하고 그것 위에 호빙할 수도 있다. 엘리먼트 존재는 위에서 설명된 기법들을 사용하여 엘리먼트가 스크린 상에 존재하는지의 여부를 체크할 수도 있다. 일부 실시예들에서, 설계자(310)에서 구현될 수 있는 수백 또는 심지어 수천 개의 활동들이 있을 수도 있다. 그러나, 임의의 수 및/또

는 유형의 활동들은 발명의 범위로부터 벗어나는 일 없이 이용 가능할 수 있다.

[0035] UI 자동화 활동들(330)은 하위 레벨 코드(예컨대, CV 활동들)로 작성되고 스크린과의 상호작용들을 용이하게 하는 특수한, 하위 레벨 활동들의 서브세트이다. UI 자동화 활동들(330)은 로봇이 원하는 소프트웨어와 상호작용하는 것을 허용하는 드라이버들(340)을 통해 이들 상호작용들을 용이하게 한다. 예를 들면, 드라이버들(340)은 OS 드라이버들(342), 브라우저 드라이버들(344), VM 드라이버들(346), 엔터프라이즈 애플리케이션 드라이버들(348) 등을 포함할 수도 있다.

[0036] 드라이버들(340)은 후크들을 찾고 키들을 모니터링하는 등 낮은 레벨에서 OS와 상호작용할 수도 있다. 이들은 Chrome®, IE®, Citrix®, SAP® 등과의 통합을 용이하게 할 수도 있다. 예를 들면, "클릭" 활동은 드라이버들(340)을 통해 이들 상이한 애플리케이션들에서 동일한 역할을 수행한다.

[0037] 도 4는 본 발명의 일 실시예에 따른 RPA 시스템(400)을 예시하는 아키텍처 도면이다. 일부 실시예들에서, RPA 시스템(400)은 도 1 및/또는 도 2의 RPA 시스템들(100 및/또는 200)일 수도 있거나 또는 이들을 포함할 수도 있다. RPA 시스템(400)은 로봇들을 실행하는 다수의 클라이언트 컴퓨팅 시스템들(410)을 포함한다. 컴퓨팅 시스템들(410)은 자신들 상에서 실행하고 있는 웹 애플리케이션을 통해 수행자 컴퓨팅 시스템(420)과 통신할 수 있다. 수행자 컴퓨팅 시스템(420)은, 결국, 데이터베이스 서버(430) 및 옵션적인 인덱서 서버(440)와 통신할 수 있다.

[0038] 도 1 및 도 3에 관해, 웹 애플리케이션이 이들 실시예들에서 사용되지만, 임의의 적합한 클라이언트/서버 소프트웨어는 발명의 범위로부터 벗어나는 일 없이 사용될 수 있다는 점에 주의해야 한다. 예를 들면, 수행자는 클라이언트 컴퓨팅 시스템들 상의 비 웹 기반 클라이언트 소프트웨어 애플리케이션들과 통신하는 서브측 애플리케이션을 실행할 수도 있다.

[0039] 도 5는 본 발명의 일 실시예에 따른, RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하도록 구성되는 컴퓨팅 시스템(500)을 예시하는 아키텍처 도면이다. 일부 실시예들에서, 컴퓨팅 시스템(500)은 본 개시에서 묘사되며 및/또는 설명되는 컴퓨팅 시스템들 중 하나 이상일 수도 있다. 컴퓨팅 시스템(500)은 정보를 통신하기 위한 버스(505) 또는 다른 통신 메커니즘과, 버스(505)에 커플링되어 정보를 프로세싱하는 프로세서(들)(510)를 포함한다. 프로세서(들)(510)는 중앙 프로세싱 유닛(Central Processing Unit; CPU), 주문형 집적회로(Application Specific Integrated Circuit; ASIC), 필드 프로그램가능 게이트 어레이(Field Programmable Gate Array; FPGA), 그래픽 프로세싱 유닛(Graphics Processing Unit; GPU), 이들의 다중 인스턴스들, 및/또는 이들의 임의의 조합을 포함하는 임의의 유형의 일반 또는 특정 목적 프로세서일 수도 있다. 프로세서(들)(510)는 다수의 프로세싱 코어들 또한 가질 수도 있고, 코어들 중 적어도 일부는 특정 기능들을 수행하도록 구성될 수도 있다. 다중 병렬 프로세싱이 일부 실시예들에서 사용될 수도 있다. 특정 실시예들에서, 적어도 하나의 프로세서(들)(510)는 생물학적 뉴런들을 흉내내는 프로세싱 엘리먼트들을 포함하는 신경모방(neuromorphic) 회로일 수도 있다. 일부 실시예들에서, 신경모방 회로들은 폰 노이만 컴퓨팅 아키텍처의 전형적인 컴포넌트들을 요구하지 않을 수도 있다.

[0040] 컴퓨팅 시스템(500)은 프로세서(들)(510)에 의해 실행될 정보 및 명령들을 저장하기 위한 메모리(515)를 더 포함한다. 메모리(515)는 임의의 조합의 랜덤 액세스 메모리(Random Access Memory; RAM), 판독 전용 메모리(Read Only Memory; ROM), 플래시 메모리, 캐시, 자기 또는 광학적 디스크와 같은 정적 스토리지, 또는 임의의 다른 유형들의 비일시적 컴퓨터 판독가능 매체 또는 그 조합들로 구성될 수 있다. 비일시적 컴퓨터 판독가능 매체는 프로세서(들)(510)에 의해 액세스될 수 있고 휘발성 미디어, 비휘발성 미디어, 또는 둘 다를 포함할 수도 있는 임의의 가용 매체일 수도 있다. 그 매체들은 또한 탈착식, 비탈착식, 또는 둘 다일 수도 있다.

[0041] 덧붙여, 컴퓨팅 시스템(500)은 무선 및/또는 유선 연결을 통해 통신 네트워크에 대한 액세스를 제공하는 통신 디바이스(520), 이를테면 트랜시버를 포함한다. 일부 실시예들에서, 통신 디바이스(520)는 본 발명의 범위로부터 벗어나는 일 없이 주파수 분할 다중 접속(Frequency Division Multiple Access; FDMA), SC-FDMA(Single Carrier FDMA), 시분할 다중 접속(Time Division Multiple Access; TDMA), 코드 분할 다중 접속(Code Division Multiple Access; CDMA), 직교 주파수 분할 다중화(Orthogonal Frequency Division Multiplexing; OFDM), 직교 주파수 분할 다중 접속(Orthogonal Frequency Division Multiple Access; OFDMA), GSM(Global System for Mobile) 통신들, 일반 패킷 무선 서비스(General Packet Radio Service; GPRS), 유니버셜 이동 통신 시스템(Universal Mobile Telecommunications System; UMTS), cdma2000, 광대역 CDMA(W-CDMA), 고속 다운링크 패킷 접속(High-Speed Downlink Packet Access; HSDPA), 고속 업링크 패킷 접속(High-Speed Uplink Packet Access; HSUPA), 고속 패킷 접속(High-Speed Packet Access; HSPA), LTE(Long Term Evolution), LTE-A(LTE Advanced),

802.11x, Wi-Fi, Zigbee, 초광대역(UWB), 802.16x, 802.15, 홈 노드-B(HnB), 블루투스, 무선 주파수 식별(RFID), IrDA(Infrared Data Association), 근접장 통신들(NFC), 5세대(5G), 뉴 라디오(New Radio; NR), 이들의 임의의 조합, 및/또는 임의의 다른 현재 존재하거나 또는 장래에 구현되는 통신 표준 및/또는 프로토콜을 사용하도록 구성될 수도 있다. 일부 실시예들에서, 통신 디바이스(520)는 본 발명의 범위로부터 벗어나는 일 없이 단일, 어레이, 위상, 스위칭, 빔포밍, 빔조향(beamsteering), 이들의 조합, 및 또는 임의의 다른 안테나 구성인 하나 이상의 안테나를 포함할 수도 있다.

[0042] 프로세서(들)(510)는 디스플레이(525), 이를테면 플라즈마 디스플레이, 액정 디스플레이(Liquid Crystal Display; LCD), 발광 다이오드(Light Emitting Diode; LED) 디스플레이, 전계 방출 디스플레이(Field Emission Display; FED), 유기 발광 다이오드(Organic Light Emitting Diode; OLED) 디스플레이, 플렉시블 OLED 디스플레이, 플렉시블 기판 디스플레이, 프로젝션 디스플레이, 4K 디스플레이, 고품위 디스플레이, 망막® 디스플레이, IPS(In-Plane Switching) 디스플레이, 또는 사용자에게 정보를 디스플레이하기 위한 임의의 다른 적합한 디스플레이에 버스(505)를 통해 추가로 커플링된다. 디스플레이(525)는 저항성, 용량성, 표면 음향 파(surface-acoustic wave; SAW) 용량성, 적외선, 광학적 이미징, 분산적 신호 기술, 음향 펄스 인식, 좌절된 내부 전반사(frustrated total internal reflection) 등을 사용하여 터치(햅틱) 디스플레이, 3차원(3D) 터치 디스플레이, 멀티 입력 터치 디스플레이, 멀티 터치 디스플레이 등으로서 구성될 수도 있다. 임의의 적합한 디스플레이 디바이스 및 햅틱 I/O는 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다.

[0043] 키보드(530)와 커서 제어 디바이스(535), 이를테면 컴퓨터 마우스, 터치패드 등은 사용자가 컴퓨팅 시스템(500)와 인터페이싱하는 것을 가능하게 하기 위해 버스(505)에 추가로 커플링된다. 그러나, 특정 실시예들에서, 물리적 키보드 및 마우스가 존재하지 않을 수도 있고, 사용자는 디스플레이(525) 및/또는 터치패드(도시되지 않음)를 통해 단독으로 디바이스와 상호작용할 수도 있다. 입력 디바이스들의 임의의 유형 및 조합이 설계 선택 사항으로서 사용될 수도 있다. 특정 실시예들에서, 물리적 입력 디바이스 및/또는 디스플레이가 존재하지 않는다. 예를 들면, 사용자는 통신하는 다른 컴퓨팅 시스템을 통해 원격으로 컴퓨팅 시스템(500)과 상호작용할 수도 있거나, 또는 컴퓨팅 시스템(500)은 자율적으로 동작할 수도 있다.

[0044] 메모리(515)는 프로세서(들)(510)에 의해 실행될 때 기능을 제공하는 소프트웨어 모듈들을 저장한다. 그 모듈들은 컴퓨팅 시스템(500)을 위한 운영 체제(540)를 포함한다. 그 모듈들은 본 개시에서 설명되는 프로세스들의 전체 또는 부분 또는 그것의 파생물들을 수행하도록 구성되는 RPA 로봇 검증 모듈(545)을 더 포함한다. 컴퓨팅 시스템(500)은 추가적인 기능을 포함하는 하나 이상의 추가적인 기능적 모듈(550)을 포함할 수도 있다.

[0045] 본 기술분야의 통상의 기술자는 "시스템"이 본 발명의 범위로부터 벗어나는 일 없이 서버, 내장형 컴퓨팅 시스템, 개인용 컴퓨터, 콘솔, 개인 정보 단말기(personal digital assistant; PDA), 셀 전화기, 태블릿 컴퓨팅 디바이스, 양자 컴퓨팅 시스템, 또는 임의의 다른 적합한 컴퓨팅 디바이스, 또는 디바이스들의 조합으로서 구현될 수 있다는 것을 이해할 것이다. 위에서 설명된 기능들을 "시스템"에 의해 수행되는 것으로서 제시하는 것은 본 발명의 범위를 어떠한 방법으로도 제한할 의도는 아니며, 본 발명의 많은 실시예들 중 하나를 제공하려는 의도이다. 실제로, 본 개시물에서 개시되는 방법들, 시스템들, 및 장치들은 클라우드 컴퓨팅 시스템들을 포함하여, 컴퓨팅 기술과 일치하는 국부적이고 분산된 형태들로 구현될 수도 있다.

[0046] 이 명세서에서 설명되는 시스템 특징들의 일부는 이들의 구현 독립성을 더 크게 강조하기 위하여 모듈들로서 제시되었다는 것에 주의해야 한다. 예를 들어, 모듈이 맞춤형 VLSI(very large scale integration) 회로들 또는 게이트 어레이들, 로직 칩들과 같은 기성(off-the-shelf) 반도체들, 트랜지스터들, 또는 다른 개별 부품들을 포함하는 하드웨어 회로로서 구현될 수도 있다. 모듈이 필드 프로그램가능 게이트 어레이들, 프로그램가능 어레이 로직, 프로그램가능 로직 디바이스들, 그래픽 프로세싱 유닛들 등과 같은 프로그램가능 하드웨어 디바이스들로 또한 구현될 수도 있다.

[0047] 모듈이 다양한 유형들의 프로세서들에 의한 실행을 위해 적어도 부분적으로는 소프트웨어로 또한 구현될 수도 있다. 식별된 실행가능 코드 유닛이, 예를 들면, 오브젝트, 절차, 또는 함수로서 편성될 수도 있는, 예를 들면, 컴퓨터 명령들의 하나 이상의 물리적 또는 논리적 블록을 포함할 수도 있다. 그럼에도 불구하고, 식별된 모듈의 실행가능물(executable)들은 물리적으로 함께 위치될 필요는 없지만, 논리적으로 함께 연결될 때, 모듈을 포함하고 그 모듈에 대한 언급된 목적을 달성하는 상이한 로케이션들에 저장된 이질적인 명령들을 포함할 수도 있다. 게다가, 모듈들은 본 발명의 범위로부터 벗어나는 일 없이 데이터를 저장하는데 사용되는, 예를 들면, 하드 디스크 드라이브, 플래시 디바이스, RAM, 테이프, 및/또는 임의의 다른 이러한 비일시적 컴퓨터 판독가능 매체일 수도 있는 컴퓨터 판독가능 매체 상에 저장될 수도 있다.

[0048] 실제로, 실행가능 코드 모듈이 단일 명령, 또는 다수의 명령들일 수도 있고, 심지어, 상이한 프로그램들 중에서, 그리고 여러 메모리 디바이스들에 걸쳐 여러 상이한 코드 세그먼트들을 통해 분산될 수도 있다. 마찬가지로, 운용 데이터(operational data)는 본 명세서에서는 모듈들 내에서 식별되고 예시될 수도 있고, 임의의 적합한 형태로 구현되고 임의의 적합한 유형의 데이터 구조 내에서 조직될 수도 있다. 운용 데이터는 단일 데이터 세트로서 수집될 수도 있거나, 또는 상이한 저장 디바이스들에 걸쳐를 포함하여 상이한 로케이션들에 걸쳐 분산될 수도 있고, 시스템 또는 네트워크 상의 단지 전자 신호들로서 적어도 부분적으로 존재할 수도 있다.

[0049] 도 6a 및 도 6b는 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 위한 프로세스(600)를 예시하는 흐름도들이다. 키 누름 기록(602)과 스크린 기록(604)(예컨대, 스크린샷들을 기록하는 것, 잠재적으로는 서버에 의한 나중의 프로세싱을 위한 더 긴 일련의 스크린샷들의 스크린 비디오 기록을 실시간으로 비교하는 것 등)은 로봇에 의해 "눌러졌던" 키들(예컨대, 로봇에 의해 컴퓨팅 시스템에서 유발되는 키 누름 이벤트들, 애플리케이션 레벨 API 호들 등) 및 그래픽 변경들이 발생했던 스크린 상의 로케이션(들)을 각각 결정하기 위해 수행된다. 그러나, 일부 실시예들에서, 로봇이 입력하고자 하였던 텍스트가 로봇의 활동들로부터 결정될 수도 있기 때문에 키 누름 기록은 수행되지 않는다. 로봇에 의해 입력된 문자들이 본 개시에서 "키 누름들"이라 불리고, 단계 602가 "키 누름 기록"이라고 불리지만, RPA 로봇은 물리적 키보드 상에서 실제로 타이핑하지 않는다. 오히려, RPA 로봇은 키보드 상에 물리적으로 타이핑하는 사용자로 인해 애플리케이션, 운영 체제, 또는 둘 다에 의해 발생될 것들과 동일하거나 또는 유사한 시스템 및/또는 애플리케이션 레벨 이벤트들을 생성한다. 일부 실시간 또는 거의 실시간 실시예들에서, 비디오 데이터는 기록 및 저장되지 않을 수도 있는데, 프레임 차이들이 즉석에서 계산될 수도 있기 때문이다.

[0050] 키 누름 기록(602)은 운영 체제로부터 키 누름 이벤트들을 모니터링함으로써 수행될 수도 있다. 그러나, 임의의 프로그래밍 언어 및 임의의 운영 체제(예컨대, 모바일, PC, Mac 등)를 위해 키 누름 정보를 제공하는 임의의 키 누름 이벤트 또는 기능은 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다.

[0051] 키 누름 이벤트는 눌러졌던 키에 연관되는 문자(예컨대, 글자 "a", 숫자 "2", "%" 부호 등), 키 누름 이벤트가 발생했던 시간 등에 관한 정보를 포함할 수도 있다. 로봇에 의해 눌려졌던 때와 대응하는 문자가 스크린 상에 나타난 때 사이의 지연을 고려하여 키 문자들의 큐(예컨대, 선입선출(FIFO) 큐)가 키가 시간 윈도우(time window)(예컨대, 500 밀리초(ms), 1 초 등) 동안 저장될 수도 있다. 이 시간 윈도우는 로봇이 키를 누른 때와 키가 스크린 상에 나타난 때 사이의 전형적인 시간 지연보다 통상적으로 더 길다(예컨대, 50 ms 문자 나타남 지연과 500 ms 버퍼 윈도우이다).

[0052] 큐는 고속의 현대 컴퓨팅 하드웨어로 인해 로봇들이 동작하는 속력으로 인해 종종 발생하는 바와 같이, 스크린 상에 한꺼번에 나타나는 다수의 문자들을 캡처할 목적으로서 또한 서빙할 수도 있다. 예를 들면, 로봇이 매우 빠르게 연속적으로 (예컨대, 1 ms 내에) "abc"를 누르지만 초당 25 개 프레임들(즉, 40 ms마다 하나의 프레임)만이 캡처되면, 텍스트 "abc"는 다음의 스크린샷에서 한꺼번에 나타날 수도 있다. 큐에 "a", "b", 및 "c"를 가짐으로써, 텍스트 인식이 이들 문자들 및/또는 시퀀스들을 찾을 때 알고리즘은 이들 문자들 및/또는 이들의 시퀀스들 각각을 서치할 수도 있다. 예를 들면, 일부 실시예들에서, 로봇이 "abc"를 입력하고 "ab"가 다음 프레임에서 나타나면, 키 누름 큐에서의 순서는 스크린 상에 나타난 순서와 동일한 것으로 가정될 수도 있다.

[0053] 특정 실시예들에서, 로봇은 키 누름들 사이에 지연을 제공할 수도 있다. 예를 들면, 로봇은 키 누름 이벤트를 생성하며, 5 ms를 기다리며, 다른 키 누름 이벤트를 생성하는 등을 할 수도 있다. 이는 로봇으로 하여금 컴퓨팅 시스템과의 더 많은 사용자 유사 상호작용들을 갖도록 할 수도 있고, 하나의 스크린샷에서부터 다른 스크린샷으로의 변경들이 덜 극단적이고 인간 상호작용 속력들에 더 가까울 수도 있기 때문에 또한 검증에 도움이 될 수도 있다. 부가적으로 또는 대안적으로, 로봇은 변경들이 제출(submission) 후 사라지기 전에 그들 변경들이 검증을 위한 스크린샷에서 나타나는 것을 허용하기 위하여 텍스트를 입력하고 있는 것과 제출 버튼, 이메일 전송 버튼 등을 "누르는 것" 사이에 충분한 시간을 기다릴 수도 있다.

[0054] 스크린 기록(604)은 발명의 범위로부터 벗어나는 일 없이 임의의 적합한 하드웨어, 소프트웨어, 또는 이들의 임의의 조합에 의해 수행될 수도 있다. 예를 들면, 비디오 기록은 외부 비디오 카메라, 내부 그래픽 카드, (연결된 컴퓨팅 시스템을 통해 또는 그런 일 없이) 디스플레이로부터의 비디오 스트림을 모니터링하는 원격 컴퓨팅 시스템 등에 의해 수행될 수도 있다. 기록된 스크린샷들은 화소가 완전한지의 여부에 관계없이 임의의 원하는 포맷(예컨대, JPEG, BMP, PNG 등)으로 저장될 수도 있다.

[0055] 하나의 구현예에서, 스크린샷들은 Format16bppRgb555 포맷에서 16비트 컬러 깊이의 BitmapData로서 저장된다. 스크린샷들을 손실 포맷으로 압축하면 정확도가 떨어질 수도 있는데, 하나 이상의 화소에서의 변경들이 일부 포

맷들에서 전체 이미지에 전파되는 컬러 맵을 변경시키거나, 컬러 깊이에 영향을 미치거나, 전체 디테일을 감소/변화시키거나, 또는 (예컨대, JPEG 포맷에서) 압축 아티팩트들의 존재에 의해 이미지 화소들에 영향을 미칠 수도 있기 때문이다. 이를 수용하기 위해, 허용오차들이 사용될 수도 있다. 스크린 기록(604)은 프레임들을 특정한 프레임 레이트로 생성하는데, 이 프레임 레이트는 컴퓨팅 시스템의 현재 프로세싱 부하(예컨대, 초당 30개 프레임들)에 의존하여 가변할 수도 있다.

[0056] 그러면 현재 스크린샷 프레임과 바로 이전 프레임은 606에서 이들 사이에 차이들이 존재하는지(예컨대, 그 프레임들 안의 대응하는 화소들 중 적어도 일부가 서로 상이한지)의 여부를 결정하기 위해 비교된다. 로봇이 워크플로우에서 취하고 있는 현재 액션(들)에 기초하여, 대응하는 스크린 변경들이 예상될 수도 있거나 또는 예상되지 않을 수도 있다. 스크린 변경이 608에서 발생하지 않았고 이것이 워크플로우 실행에 기초하여 610에서 예상된 경우, 프로세스는 612에서 다음의 스크린샷으로 진행하고 606에서 이 다음의 스크린샷과 이전의 현재 스크린샷을 비교하는 것으로 리턴한다.

[0057] 스크린 변경이 608에서 발생하지 않았고 610에서 변경이 예상되었다면, 또는 스크린 변경이 608에서 일어났지만 614에서 변경이 예상되지 않았다면, 시스템은 예외를 발생시키거나 또는 로봇은 616에서 시정 조치를 취할 것이 지시된다. 예외를 둔 경우, 로봇의 동작은 정지될 수도 있고 로봇이 원하는 액션(들)을 달성하는데 실패했음을 나타내는 메시지는 스크린 상에 디스플레이되거나 또는 컴퓨팅 시스템의 사용자에게 (예컨대, 텍스트 또는 이메일을 통해) 전송될 수도 있다. 특정 실시예들에서, 실패했던 액션(들)은 사용자에게 제공될 수도 있으며 그래서 사용자는 문제를 해결하려고 시도할 수 있거나, 또는 사용자는 RPA 개발자에 의해 수정될 문제를 보고할 수 있다.

[0058] 예외를 발생시키는 대신, 다른 옵션은 로봇이 시정 조치를 취하게 하는 것이다. 로봇은 워크플로우의 단계(들)가 실패하였음을 알 수 있으며, 얼마간의 수정 조치를 취하고, 단계(들)를 다시 시도할 수도 있다. 이는 로봇이 드라이버 레벨에서 로봇이 상호작용하려고 시도하고 있는 애플리케이션용 윈도우가 액티브 윈도우(active window)인지의 여부를 체크하는 것, 관련 있는 그래픽 엘리먼트에 대한 마우스 포인터 또는 캐럿의 로케이션을 체크하는 것, 올바른 그래픽 엘리먼트가 액티브 엘리먼트인지의 여부를 체크하는 것, 이들의 임의의 조합 등을 비제한적으로 포함할 수도 있다. 로봇은 올바른 윈도우를 액티브의, 포커싱된 윈도우로 설정하며, 원하는 액티브 엘리먼트로 포커스를 설정하며, 마우스를 이동시키고 클릭 등을 한 다음 단계(들)를 다시 시도할 수도 있다. 시정 조치가 성공되면, 프로세스는 단계 612로 리턴할 수도 있다. 그러나, 시정 조치가 실패하면, 예외를 둘 수도 있다.

[0059] 만일 스크린 변경이 608에서 일어났고, 그 변경이 워크플로우에 기초하여 614에서 예상되었다면, 프로세스는 도 6b로 진행한다. 스크린이 변경된 영역들은 결정되고 그 영역들에서의 문자들은 (만약 있다면) 618에서 결정된다. 특정 실시예들에서, 이는 원하는 윈도우가 나타났거나 또는 사라졌음을 결정하는 것, 이미지가 나타났거나 또는 변경되었음을 결정하는 것, 그래픽 컴포넌트(예컨대, 텍스트 박스, 텍스트 필드, 테이블 등)가 나타났거나 또는 변경되었음을 결정하는 것, 이들의 조합 등을 포함할 수도 있다. 이러한 변경들은, 예를 들어, CV를 사용하여 결정될 수도 있다.

[0060] 특정 실시예들에서, 다수의 프레임들이 정확도를 추가로 증가시키는데 사용될 수도 있다. 예를 들면, 문자가 입력되는 때와 그 문자가 스크린 상에 나타나는 때 사이에 가변하는 (예컨대, 30 ms부터 42 ms로 24 ms로 등으로 가변하는) 비동기화가 있다면, 다수의 프레임들을 사용하는 것이 입력된 텍스트를 식별하는데 도움이 될 수도 있다. 그 다음에 시각적 변경들의 로케이션(들)은 격리될 수도 있고, 문자들을 인식하기 위해 변경이 발생했던 로케이션 상에서 알고리즘이 실행된다. 이 알고리즘은 OCR, 특정한 폰트들에서의 문자들의 부울 어레이 맵들에 대한 화소 영역 비교들을 등을 사용할 수도 있다. 일부 실시예들에서, 문자 인식은 발생된 변경이 격리되고 분석되는 상대적으로 작은 영역들 상에서만 실행되고, 나머지 영역들은 폐기되다. 이는 전체 스크린(예컨대, 3840×2160 화소 해상도) 상에서 OCR을 실행하는 것이 문자들이 스크린 상에 나타나는 속력을 컴퓨팅 시스템이 따라잡기에는 너무 컴퓨테이션적으로 비쌀 수도 있는 컴퓨팅 시스템들 상에서 알고리즘이 실시간으로 실행할 수 있는 것을 보장하는데 도움이 된다. 그러나, 충분한 프로세싱 파워를 갖는 컴퓨팅 시스템들의 경우, 전체 스크린은 변경이 먼저 발생하지 않았던 영역들을 파싱하는 일 없이 분석될 수도 있다.

[0061] 위와 같이, 특정 실시예들에서, 완벽한 화소가 아니라, 비디오 프레임 비교 컴퓨테이션들이 허용오차를 사용한다. 컬러 세기들, 밝기, 및/또는 다른 인자들은 이들이 특정한 허용오차 내에 있으면 동일한 것으로 간주될 수도 있다. 예를 들면, 화소들은 그들의 값들 중 하나 이상에서의 변경들이 특정한 수, 특정한 백분율 등 미만이면 동일한 것으로 간주될 수도 있다. 5 미만, 1% 미만 등의 적색, 녹색, 청색, 및 밝기에서의 변경이 동일한

것으로 간주될 수도 있다. 특정 실시예들에서, 이들 변수들 중 하나 이상은 상이한 허용오차들을 가질 수도 있다. 예를 들면, 아마도 밝기 변경들은 실제 화소 변경들을 나타내기 위해 컬러 변경들보다 더 크거나 또는 더 작아야 되는 것이 필요할 수도 있다. 퍼지 이미지 매칭은 유사성들/차이들을 식별하기 위해 특정 실시예들에서 수행될 수도 있다.

[0062] 일부 실시예들에서, 퍼지 이미지 매칭은 밝기, 이미지 템플릿들, 에지 비교들, 2치화, 다운스케일 및 비트 감소, 딜레이션(dilation)을 고려하며, 매칭들을 더 정확하게 식별하기 위해 커널 블러링, 이들의 조합 등을 적용한다. RGB 값들에 허용오차를 적용하는 화소 대 화소 RGB 매칭이 사용될 수도 있으며 그래서 정확히 동일하지 않은 가까운 값들이 매칭들로서 식별될 수도 있다. 비트 깊이 및/또는 컬러 스케일은 감소될 수도 있고 화소 대 화소 RGB 또는 그레이스케일 매칭이 적용될 수도 있다. 이미지들로부터의 에지들은 검출되고 비교될 수도 있다. 2치화는 이미지들에 적용될 수도 있고(예컨대, 이진 문턱값, Otsu 문턱값, 적응적 문턱값 등) 화소 대 화소 매칭이 이진 이미지들에 적용될 수도 있다. 이미지들의 스케일이 감소될 수도 있고 화소 대 화소 매칭이 수행될 수도 있다. 이미지들의 딜레이션이 수행될 수도 있고 그 다음에 화소 대 화소 매칭이 적용될 수도 있다. 키 포인트들은 이미지들로부터 추출될 수도 있고(예컨대, MSER(maximally stable extremal region) 디스크립터들) 추출된 키 포인트들은 특징 매처(matcher)들을 사용하여 비교될 수도 있다(예컨대, 브루트 힘(brute force) 매칭, k 최근접 이웃들(k-nearest neighbors; KNN) 매칭 등).

[0063] 공차 기반 컴퓨테이션들이 유익할 수도 있는 다양한 이유들이 있다. 예를 들면, 프레임이 캡처된 후 이미지가 압축되면, 손실 압축이 화소 값들에 영향을 미칠 수 있으므로 허용오차들이 컴퓨테이션들에 포함되어야 한다. 또한, 원래의 시각적 소스는 (예컨대, 가상 컴퓨팅 시스템이 에뮬레이터를 통해 시작되고 에뮬레이터가 가상 컴퓨터 스크린 콘텐츠를 압축할 때) 손실 압축을 사용하여 캡처하기 전에 압축될 수도 있다. 이는 이미지들이 원격 머신(예컨대, 서버)에서부터 국부 컴퓨팅 시스템으로 방송되기 때문에 발생할 수도 있다.

[0064] 변경이 발생했던 그리고/또는 다른 그래픽 변경들이 발생했던 스크린 영역(들)의 문자들이 일단 618에서 식별되면, 그 문자들은 키 누름 이벤트들에 대응하는 저장된 문자들의 큐와 비교된다. 매칭이 발견되면, 매칭 로케이션의 스크린 좌표들, 뿐만 아니라 다른 검출된 그래픽 변경들의 스크린 좌표들이 620에서 추출된다. 일부 경우들에서, 문자 인식 알고리즘은 스크린 상에서 문자를 실제로 그것이 무엇에 대한 것인지를 인식하는데 실패할 수도 있다. 예를 들면, OCR 알고리즘은 스크린 상의 글자 "O"를 숫자 "0"으로서 인식할 수도 있다. 그 경우, 일부 실시예들에서, 알고리즘은 스크린 상의 캐럿의 로케이션을 추적한다. 이는 다양한 캐럿 형상들의 이미지 패턴들과 스크린을 이미지 검출(예컨대, CV) 등을 사용하여 비교함으로써 결정될 수 있습니다 일부 실시예들에서, 알고리즘은 캐럿 깜박임을, 캐럿이 깜박인다면 고려할 수도 있다.

[0065] 특정 실시예들에서, 퍼지 매칭이 OCR 결과들과 큐에서의 문자들을 비교하는데 사용될 수도 있다. 퍼지 매칭 로직은 글자 "O"가 숫자 "0"과 비슷해 보인다는 것을 인식할 수도 있고 이들 문자들을 매칭으로서 식별할 수도 있다. 큐에 다른 유사한 문자들이 없다면, 그 매칭은 확정(confirm)될 수도 있다.

[0066] 특정 실시예들에서, 캐럿 추적이 수행된다. 이미지의 변경된 영역(들)의 분석은 캐럿에 해당할 수도 있는 후보 리스트(예컨대, 후보들이 얇은 수직 라인 또는 유사한 것으로서 나타남)를 생성하기 위해 수행될 수도 있다. 후보가 시간이 지남에 따라 깜박인다는 것을 식별하기 위해 검증이 수행될 수도 있고, 그러면 실제 캐럿은 식별될 수도 있다. 게다가 텍스트 입력을 할 수 있는 그래픽 엘리먼트(예컨대, 텍스트 박스, 워드 프로세서 문서, 텍스트 필드 등) 내에 캐럿이 나타남을 검증하기 위해 검증이 수행될 수도 있다.

[0067] 만일 스크린 상의 다른 곳에서 변경들이 발생하지 않았거나, 또는 다른 변경들이 누락된 문자 외의 큐에서의 문자들과 매칭되면, 알고리즘은 이것이 식별되지 않은 유일한 변경들이기 때문에 누락 문자이어야 한다고 유추할 수도 있다. 그 다음에 알고리즘은 인식된 글자 "O"가, 예를 들면, 문자 큐에서 실제로 달리 식별되지 않은 "0"임을 유추하고 620에서 매칭 로케이션의 스크린 좌표들을 추출할 수도 있다. 이는 알고리즘의 정확도를 개선할 수도 있다.

[0068] 일부 실시예들에서, 문자들은 미리 결정된 시간(예컨대, 500 ms, 1초 등)이 경과된 후, 문자들 또는 문자 시퀀스가 스크린 상에서 발견되고 독특하게 식별될 때 큐로부터 제거될 수도 있으며, 미리 결정된 사이즈(예컨대, 20 개 문자들)의 큐에 기초하여 큐의 끝에서 문자들이 사라지게 하는 등을 한다. 시간 윈도우를 벗어나는 문자들을 제거하기 위하여, 큐는 입력되었던 문자 및 타임 스탬프를 갖는 변수들을 저장할 수도 있다. 알고리즘은 큐에서의 (잠재적으로 "선입(first in)" 단에서 시작하는) 키 누름 변수들의 타임 스탬프들을 현재 시간과 주기적으로 비교할 수도 있다. 시간 윈도우보다 더 오래된 키 누름 변수가 큐에서 발견되면, 그 변수는 제거될 수도 있다. 특정 실시예들에서, 일단 시간 윈도우 내에 있는 키 누름 변수가 발견되면, 큐에서의 모든 다른 변수

들이 시간 윈도우 내에 있다고 가정될 수도 있고, 프로세싱은 해당 반복에 대해 중지될 수도 있다.

[0069] 최근에 입력된 문자들 및/또는 식별된 그래픽 엘리먼트들 내의 스크린 영역의 좌표들이 620에서 추출된 후, 622에서 그 좌표들은 실행중인 애플리케이션(들)과 비교되고 액티브 엘리먼트는 추출된 좌표들이 속하는 엘리먼트에 기초하여 결정된다. 좌표들이 그래픽 애플리케이션에서의 임의의 잠재적 액티브 엘리먼트들과 대응하지 않거나, 또는 그 좌표들이 잘못된 액티브 엘리먼트와 대응하는 것이 가능하다. 입력된 텍스트가 불완전하거나 또는 잘못되거나, 또는 원하는 스크린 변경이 발생하지 않았던 것이 또한 가능하다. 따라서, 워크플로우에서 대응하는 로봇 활동 또는 활동들을 체크하고 액티브 엘리먼트, 입력된 텍스트, 및/또는 다른 원하는 스크린 변경들이 발생하였음을 보장함으로써 검증이 624에서 수행된다. 이는 로봇이 대응하는 활동 또는 일련의 활동들(예컨대, 필드를 선택하는 것, 그 필드에 텍스트를 입력하는 것, 버튼을 클릭하는 것 등)에서 취할 액션을 결정하는 것, 활동 또는 일련의 활동들(예컨대, 새로운 애플리케이션 윈도우가 열릴 것으로 예상되는 것, 애플리케이션 윈도우가 새로운 포맷으로 변경할 것으로 예상되는 것, 특정한 텍스트가 애플리케이션에 의해 제시될 것으로 예상되는 것 등)에 기초하여 스크린 변경이 발생해야 한다고 결정하는 것, 또는 발명의 범위로부터 벗어나는 일 없는 임의의 다른 적합한 검증 액션(들)을 비제한적으로 포함할 수도 있다. 검증은 로봇 자체에 의해, 로봇의 실행을 모니터링하는 다른 애플리케이션 또는 로봇에 의해, 또는 둘 다에 의해 수행될 수도 있다. 검증이 626에서 성공하면, 프로세스는 도 6a의 단계 612에서 다음 스크린샷으로 진행한다. 그러나, 검증이 실패하면(예컨대, 로봇이 "$1,234.50"을 인보이스를 위한 필드에 입력해야 했지만, 대신 이것이 "회사 이름" 필드에서 나타났다면, 또는 문자가 누락되었기 때문에 입력된 값이 잘못되었다면(예컨대, "$1,34.50")), 프로세스는 도 6a의 단계 616에서 시정 조치를 취하는 예외를 발생시키는 것으로 진행한다.

[0070] 일부 실시예들에서, 스크린샷 기록, 키 누름 기록, 및/또는 그것의 프로세싱은 RPA 로봇에 의해 수행된다. 특정 실시예들에서, 레코더 애플리케이션이 로봇 활동을 스크린샷들 또는 비디오로서 기록하며, 일련의 키 누름들을 기록하고, 이 정보를 나중의 프로세싱을 위해 저장하거나 또는 이 정보를 컴퓨팅 시스템 또는 다른 컴퓨팅 시스템 상에서 실행하고 있는 다른 애플리케이션에 실시간 또는 거의 실시간 프로세싱을 위해 전달한다. 경계 직사각형(bounding rectangle)들을 포함하는 인식된 그래픽 엘리먼트들의 세트를 제공하기 위해, CV는 비디오 및 키 누름들의 프로세싱에 앞서, 비디오 및 키 누름들의 프로세싱 직후에 적용되거나, 또는 나중에 적용될 수도 있다. 그 다음에, 그래픽 엘리먼트 경계 직사각형과 캐럿/텍스트의 좌표들 사이에서 교차점이 발견되면, 특정 엘리먼트는 현재 액티브, 또는 "포커싱된"(즉, "액티브 엘리먼트") 것으로 간주될 수도 있다.

[0071] 일부 경우들에서, 스크린에 대한 하나의 프레임에서 다음 프레임으로의 변경들은 상당할 수도 있다. 예를 들면, 로봇이 윈도우를 닫을 때, 스크린의 대부분은 변경될 수도 있다. 그러므로, 일부 실시예들에서, 변경 문턱값이 결정되고 여하튼 (예컨대, 스크린의 2%, 5%, 10%, 30%, 50%, 70% 등을 초과하여 변경된) 시간 인접 스크린샷들을 비교할지의 여부를 결정하기 위해 적용된다. 이 문턱값이 충족되거나 또는 초과될 때, 두 개의 시간 인접 프레임들 사이의 차이가 문턱값 미만이 되기까지 프레임 비교 프로세스는 스킵될 수도 있다.

[0072] 일부 실시예들에서, 로봇은 확정이 수신되기까지 컴퓨팅 시스템의 스크린에 대한 변경들을 유발하는 워크플로우 단계들 후에 일시 중지될 수도 있다. 이는 이전의 단계가 실패했을 때 로봇이 다음 단계로, 그 다음으로, 그리고 그 다음으로 실행하는 것을 방지한다. 따라서, 잘못된 동작이 빠르게 식별되고 중단될 수도 있다.

[0073] 모니터링 및 로봇 검증 프로세스는 일부 실시예들에서 컴퓨팅 시스템을 제어하려고 시도하는 동일한 로봇에 의해 수행될 수도 있다. 이는 로봇이 자신 소유의 동작을 검증하는 메커니즘을 제공한다. 특정 실시예들에서, 동일한 컴퓨팅 시스템 상에서 동작하거나, 또는 비디오 및 키 누름 정보를 원격으로 수신하는 상이한 컴퓨팅 시스템(예컨대, 서버) 상에서 동작하는 다른 로봇이 모니터링 및 검증을 수행할 수도 있다. 일부 실시예들에서, 국부 또는 원격으로 동작하는 비 로봇 애플리케이션이 모니터링 및 검증을 수행한다. 특정 실시예들에서, 비디오 및 키 누름들의 분석을 수행하는 애플리케이션 또는 로봇과 검증을 수행하는 로봇 또는 애플리케이션이 상이하다.

[0074] 도 7은 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 위한 프로세스(700)를 예시하는 아키텍처 도면이다. 스크린 기록(710)이 프레임 N(712), 프레임 N - 1(714), 프레임 N - 2(716), 및 프레임 N - 3(718)을 생성하도록 수행된다. 프레임들은 전체 스크린, 러닝 애플리케이션에 연관된 윈도우 등일 수도 있다. 특정 실시예들에서, 다수의 실행중인 윈도우식 애플리케이션들에 대한 프레임 세트들이 저장 및 비교될 수도 있거나 또는 이들 프레임들은 스크린샷들로부터 꺼내어질 수도 있다. 키 누름 이벤트 기록 또는 로봇 활동 텍스트 리트리벌(retrieval)(720)이 로봇에 의해 시도된 키 누름들에 연관되는 문자들(732)을 포함하는 시간 윈도우식 키 문자 큐(key character queue)(730)를 생성하기 위해 또한 수행된다. 일부 실시예들에서, 문자

들은 이들에 연관된 키 누름이 미리 결정된 연령을 초과할 때 문자 큐로부터 제거된다. 특정 실시예들에서, 문자들은 이들이 고유하면 스크린 상에서 발견된 후 큐로부터 제거된다. 그러나, 고유하지 않으면(예컨대, 사용자가 "a a a a a a"를 연속하여 빠르게 누르면), "a"의 가장 오래된 인스턴스가 일부 실시예들에서 큐로부터 제거될 수도 있다.

[0075] 코어 로직(740)은 현재 프레임과 바로 이전 프레임(이 경우, 프레임들(712, 714)), 뿐만 아니라 키 문자 큐(730)를 수신한다. 코어 로직(740)은 일부 실시예들에서 도 6a 및 도 6b에 관해 위에서 설명된 로직의 일부 또는 전부를 수행할 수도 있다. 예를 들면, 코어 로직(740)은 프레임들(712, 714)을 서로 비교하고 시각적 변경들이 발생했던 영역(들)을 프레임 안에서 결정할 수도 있다. 이는, 예를 들어, 각각의 화소들의 적색/녹색/청색(RGB) 값들이 서로에 대한 문턱값을 초과하는지의 여부를 비교함으로써 결정될 수도 있다.

[0076] 코어 로직(740)은 시각적 변경들의 로케이션(들)을 분리하며, 로케이션(들)에서 문자들을 인식하기 위해 문자 인식을 수행하고, 예를 들어, 문자들이 나타날 것으로 예상되었던 것들에 대응하는지의 여부를 키 문자 큐(730)의 콘텐츠들에 기초하여 결정할 수도 있다. 코어 로직(740)은 인식된 문자들을 키 문자 큐(730)에서의 문자들(732)에 또한 매칭시킬 수도 있다. 매칭들이 발견될 때, 매칭 로케이션의 스크린 좌표들은 추출되고 캐럿 및/또는 타이핑 영역들(750)로서 제공될 수도 있다.

[0077] 문자 인식을 사용하는 것에 더하여 또는 그것에 대안적으로, 일부 실시예들에서, 이미지 인식(예컨대, CV)이 새로이 나타나는 또는 사라지는 이미지들 또는 아이콘들을 검출하는데 사용될 수도 있다. 그 프로세스는 그렇지 않으면, 영역 차이 분석 스테이지에서의 이 대체 또는 보충적 로직을 제외하면, 위에서 설명된 것과 동일할 수도 있다. 이는, 예를 들어, 로봇이 이미지들 또는 텍스트를 문서 또는 필드에 잘라내고 붙여 넣는 곳을 결정하는데 유용할 수도 있다.

[0078] 텍스트를 잘라내고 붙여 넣는 경우, 텍스트가 클립보드들로부터 플러싱(flushing)될 때, 개별 문자들은 키 누름 이벤트들로서 캡처되지 않는다. 이러한 경우, 클립보드의 콘텐츠와 사용자가 CTRL+V를 누르는 시간이 캡처될 수 있다. 그 다음에 클립보드의 콘텐츠는 개별 문자들 및 문자 시퀀스들이 식별되는 방식과 유사하게, OCR 결과들과 비교될 수 있다. 그러나, 입력된 문자 큐를 검토하는 것이 아니라, 알고리즘은 CTRL+V가 발생하기 전에 클립보드에 저장되었던 문자들의 문자열로 작업할 것이다.

[0079] 일부 실시예들에서, 현재 액티브 엘리먼트는 로봇이 스크린 변경을 유발하는 특정한 키를 누르는 경우에 저장될 수도 있다. 예를 들면, 로봇이 Enter 키를 누르면, 이는 양식(form)이 제출되게 그리고 윈도우가 닫히게 할 수도 있다. 다른 예에서, 로봇이 웹 브라우저를 위한 URL 입력 필드에서 Enter를 누르면, 이는 웹 브라우저로 하여금 웹사이트로의 내비게이션을 시작하게 한다. 알고리즘은 이를 인식하고 상당한 스크린 변경들이 발생할 가능성이 있기 때문에 이들 시나리오들에서 액티브 엘리먼트가 있는 곳을 찾기 전에 특정한 시간량을 기다릴 수도 있다. 일단 스크린이 비교적 정적이 되면(예컨대, 스크린의 상대적으로 작은 부분만이 하나의 스크린샷에서부터 다음의 스크린샷으로 변경되면), 입력된 텍스트 검출 및 캐럿 추적은 새로운 액티브 엘리먼트를 찾기 위해 다시 재개할 수도 있다. 따라서 로봇이 Enter, escape 등을 눌렀을 때 포커싱되었던 엘리먼트를 아는 것이 도움이 될 수도 있다. 또한, 현재 운영 체제가 현재 포어그라운드 윈도우(예컨대, user32.dll in Windows®에서 이용 가능한 "GetForegroundWindow")의 윈도우 경계 직사각형을 기본적으로 획득하는 신뢰성 있는 방법을 제공하면, 포어그라운드 윈도우 경계 직사각형은 스크린 캡처 및 프레임 프로세싱을 위해 사용될 영역을 제한하는데 사용될 수도 있다.

[0080] 일부 실시예들에서, 포커싱된 또는 액티브 엘리먼트가 결정되면, 텍스트로 하여금 나타나지 않게 하는 키들(예컨대, Enter, escape, control, control 및/또는 Alt 및 하나 이상의 문자의 조합 등)을 누르는 것으로부터의 거동은 액티브 엘리먼트의 특성에 기초하여 결정될 수도 있다. 예를 들면, 이러한 "핫 키"(비가시적 키 또는 키들의 조합)가 눌러질 때, "핫 키"가 트리거했던 액션은 분류될 수 있다. 포커싱된 엘리먼트가 "패스워드"로 라벨표시되고 로봇이 "Enter"를 누르면, "Enter"를 누르는 것이 "로그인" 액션을 나타낸다는 것이 유추될 수 있다. 이는 로봇이 "Enter"를 눌렀음을 그냥 아는 것보다는 더 설명적일 수도 있다.

[0081] 도 8a는 본 발명의 일 실시예에 따른, 각각 프레임 N - 1 및 프레임 N에 대한 동문 기부 윈도우를 포함하는 스크린샷들(800, 810)의 부분들(802, 812)을 예시한다. 도 8a에서 알 수 있는 바와 같이 프레임 N - 1에서의 모든 필드들은 비어 있지만, 프레임 N에서, 로봇은 이름(first name) 필드에 글자 "E"를 입력하였다. 이 변경이 발생해야 하였는지를 결정하기 위하여, 일부 실시예들의 알고리즘은 스크린샷들을 스케일링하고 이들을 정사각형들로 정규화한다. 이 경우, 스크린샷(800)(도 8b 참조)과 스크린샷(810)(도 8c 참조)은 64 개 정사각형들 곱하기 48 개 정사각형들의 그리드로 정규화되고, 이것의 27×24 부분이 도 8b 내지 도 8f에서 보여진다. 이들

정사각형들, 또는 영역들은, 64×48 이진 매트릭스로서 표현될 수도 있다.

[0082] 그 다음에 각각의 스크린샷에 대한 행은 그 행에 포함된 임의의 화소들이 변경되었는지를 알기 위해 서로 비교될 수도 있고, 매트릭스의 값들은 변경을 갖는 각각의 행마다 "1"로 설정될 수도 있다. 이는 영이 아닌 엘리먼트가 각각의 행에 존재하는지의 여부를 체크함으로써 행해질 수도 있다. 도 8d에서 알 수 있는 바와 같이, 알고리즘이 매트릭스의 본 개시에서 814가 지정된 행 4를 체크할 때, 그 안에서 변경이 검출된다. 그러면 알고리즘은 도 8e에 도시된 바와 같이 행 4로 이동하고, 열 7에서 816이 표시된 정사각형은 새로 입력된 글자 "E"의 부분을 포함한다. 알고리즘은 변경들을 포함하는 정사각형들을 계속 식별하고 이 부분 상에서 실행될 OCR을 위해 제출될 영역으로서 인접한 정사각형들(818)을 함께 스티칭한다. 도 8f를 참조한다. 이는 본 개시에서 "영역"이라 표시된다. 일부 실시예들에서, 그 영역을 구성하는 하나 또는 일부 정사각형들이 변경되지 않으면, 이들 정사각형들은, 예를 들면 OCR 프로세싱을 더 빨리 실행하기 위하여, 전적으로 하나의 컬러가 되도록 설정될 수도 있다. 특정 실시예들에서, 충분한 프로세싱 리소스들이 이용 가능하면, 스크린샷들 사이의 차이 영역들은 식별될 수도 있으며, OCR은 텍스트를 인식하기 위해 차이 영역들에 수행될 수 있고, 이 텍스트는 매칭들을 서치하기 위해 키 문자 큐와 비교될 수 있다.

[0083] "E"가 이름 필드에서 나타날 것으로 예상되었던 경우, 프로세싱 및 검증은 다음 프레임으로 진행할 수도 있다. 그러나, 글자 "E"가 올바른 엔트리가 아니었거나(예컨대, "Ed"가 "E" 대신 예상되었거나), 잘못된 필드에서 나타난 "E"이거나, 또는 둘 다이면, 이는 검증 로직에 의해 인식될 수도 있다. 그러면 로봇은 부정확한 텍스트를 제거하는 것과 동일한 필드 또는 상이한 필드 안에 올바른 텍스트를 입력하는 것 또는 일부 다른 시정 조치를 취하는 것을 시도할 수도 있다. 시정 조치가 실패하면, 로봇은 자신의 워크플로우의 실행을 일시 중지 또는 정지하거나, 사용자 또는 개발자에게 에러를 통지하는 등을 할 수도 있다.

[0084] 가끔, 디스플레이 해상도는 변경될 수도 있으며, 추가적인 모니터가 연결될 수도 있는 등이 일어날 수 있다. 일부 실시예들은 이들 변경들을 검출하고 수용하여서 검출 및 검증 프로세스들은 정확하게 유지된다. 도 9는 본 발명의 일 실시예에 따른 해상도 변경들에 대한 체크, 연결된 디스플레이들의 범위에서의 변경들에 대한 체크, 및 변경들을 수용하기 위한 프레임 캡처 로직의 설정을 위한 프로세스(900)를 예시하는 흐름도이다. 프로세스는 910에서 컴퓨팅 시스템을 위한 하나 이상의 연결된 디스플레이를 체크하는 것과 920에서 연결된 디스플레이(들)와 이전에 연결된 디스플레이(들)를 비교하는 것으로 시작한다. 이는 상이한 디스플레이 디바이스가 존재하는지의 여부를 체크하는 것, 디스플레이 디바이스의 해상도가 변경되었는지의 여부를 체크하는 것 등을 수반할 수도 있다. 일부 실시예들에서, "연결된" 디스플레이가 (예컨대, 통상적으로 스마트 폰들, 랩톱 컴퓨터들 등의 경우에서처럼) 컴퓨팅 시스템과 통합된 디스플레이일 수도 있다.

[0085] 연결된 디스플레이 디바이스(들) 및/또는 해상도가 930에서 변경되었다면, 해상도 및 스케일은 940에서 각각의 연결된 디스플레이에 대해 획득된다. 캡처될 스크린샷들을 위한 스크린샷 영역은 950에서 스케일에 의해 곱해진 그리고 원하는 사이즈(예컨대, 8, 16 등)의 배수들로 정렬된 전체 디스플레이 치수들로 설정된다. 배수들은 본 개시에서 나중에 더 상세히 논의되는 바와 같이 스크린샷들을 정사각형들로 나누는 것을 용이하게 할 수도 있다. 그 다음에 프레임 캡처 로직은 960에서 설정된다(예컨대, 재시작되며, 다시 초기화되며, 새로운 디스플레이 설정들이 제공되는 등이 된다).

[0086] 도 10은 본 발명의 일 실시예에 따른 비디오 레코더를 위한 프로세스(1000)를 예시하는 흐름도이다. 프로세스는 1010에서 스크린샷을 찍는 것으로 시작한다. 일부 실시예들에서, 이는 Windows®를 위한 그래픽 디바이스 인터페이스(Graphics Device Interface; GDI) CopyFromScreen() 명령을 사용하여 C#에서 완수될 수도 있다. 그 다음에 스크린샷은 1020에서 버퍼에 프레임으로서 추가된다. 이는, 예를 들어, C#에서 비트맵 오브젝트로서 스크린샷을 버퍼에 추가함으로써 완수될 수 있다. 프로세스가 1030에서 여전히 실행되고 있다면(예컨대, 프로세스가 애플리케이션을 닫는 것, 스크린 해상도 변경 등에 의해 중지되지 않았다면), 스크린샷 캡처는 다음의 스크린샷에 대해 반복될 수도 있다. C# 예들이 제공되지만, 본 개시에서 개시된 프로세스(1000) 및 다른 프로세스들의 경우, 임의의 적합한 프로그래밍 언어, 운영 체제, API들, 및 함수들이 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다는 것에 주의해야 한다.

[0087] 일부 실시예들은 로봇이 포커싱한 스크린의 엘리먼트를 더 정확하게 식별하기 위해 캐럿 추적을 수행한다. 예를 들면, 텍스트가 나타나고 있는 그래픽 엘리먼트에서 캐럿이 나타나면, 키 문자 큐에 새로 추가된 텍스트가 이 그래픽 엘리먼트에서 나타나는 것일 가능성이 있다. 도 11은 본 발명의 일 실시예에 따른 캐럿 추적을 수행하기 위한 프로세스(1100)를 예시하는 흐름도이다. 통상적으로, 사용자가 가장 최근에 클릭했던 로케이션에서 또는 그 로케이션 근처에서 캐럿이 나타나고 깜박이기 시작할 것이다. 따라서, 일부 실시예들은 마지막 마우스

클릭의 좌표들을 저장하고 이 로케이션에 근접한 캐럿을 서치한다. 이는 캐럿을 찾기 위해 프로세싱되는 스크린샷의 양을 감소시킬 수도 있고 정확도를 추가로 증가시킬 수도 있다. 일부 실시예들에서, 마우스 클릭들의 이력 버퍼 또는 단일의 가장 최근의 클릭 로케이션이 사용된다. 특정 실시예들에서, 예를 들어, 로봇이 탭 키를 "누르면", 시스템은 캐럿이 스크린 상의 다음 그래픽 엘리먼트로 이동할 수도 있다고 가정할 수도 있고, 해당 로케이션이 알려진다면 해당 로케이션에 서치를 재포커싱하거나, 또는 아니면 전체 스크린샷을 서치할 수도 있다.

[0088] 화소 변경들은 변경들이 발생했던 스크린샷에서의 각각의 영역에 대해 계산되고, 그 영역들은 1105에서 이진 매트릭스에 투영된다. 이진 매트릭스는 영역의 화소들이 변경하였는지의 여부의 표현이고, 스크린샷들 사이에 변경이 없는 화소들에 대한 "0"과 변경된 화소들에 대한 "1"을 포함할 수도 있다. 일부 실시예들에서 스크린샷으로부터의 다수의 정사각형들을 포함할 수도 있는 변경들이 발생했던 정사각형이 "영역"이다. 그러나, 임의의 다른 적합한 형상(예컨대, 직사각형, 육각형 등)이 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다. 일부 실시예들에서, 고정된 수의 영역들이 컴퓨팅 시스템의 프로세싱 능력에 의존하여, 분석을 위해 지원된다. 예를 들면, 일부 실시예들은 2 개의 영역들, 3 개의 영역들, 10 개의 영역들 등의 추출 및 OCR을 지원한다. 일부 실시예들은 캐럿 및 입력되거나 또는 붙여 넣은 텍스트 둘 다를 찾을 수도 있다. 수(L)를 초과하는 변경 영역들이 현재 스크린샷과 이전 스크린샷 사이에서 발견될 때, 발견되었던 처음 L 개의 영역들은 프로세싱될 수도 있거나 또는 스크린샷은 전적으로 무시될 수도 있다. 이는 사용자가 다른 윈도우를 시작했거나 또는 스크린의 충분한 부분이 달리 변경되어 다음의 스크린샷이 캡처되기 전에 OCR이 완료되지 않을 수 있는 스크린들을 무시하는데 도움이 될 수도 있다.

[0089] 각각의 이진 매트릭스에 대해, 깜박임 캐럿 영역 후보들은 1110에서 추출되고 이진 매트릭스 멤버들은 1115에서 결합된다. 본 개시에서 사용되는 바와 같이, "멤버들"은 이진 매트릭스에 존재하는 연결된 형상들, 이를테면 글자, 커서 등을 나타내는 형상들이다. 매트릭스 멤버들의 결합은, 예를 들어 컴포넌트들이 8개 연결되는, 연결된 컴포넌트들 알고리즘(Connected Components algorithm)을 사용하여 수행될 수도 있다. 연결된 컴포넌트들은 각각의 화소가 모든 다른 화소들에 연결되는 화소 세트이다.

[0090] 1120에서 결과들을 결합하는 매트릭스 멤버로부터 형상들이 추출되고 1125에서 그 형상들은 검증된다. 형상은, 예를 들어 선을 포함할 수 있는, 통상적으로 완벽한 직사각형이어야 한다. 검증된 형상 후보들은 1130에서 저장되고 확정 큐와 비교된다. 캐럿 후보들의 포지션, 사이즈, 및 모양은 타임 스탬프와 함께 저장될 수도 있다. 캐럿이 깜박이는 빈도는 허용오차(예컨대, 5%) 내에서 일정해야 한다. 캐럿이 깜박이므로, 후보들은 이들이 캐럿의 예상되는 성질들(즉, 포지션, 사이즈, 및 빈도)과 매칭되는지의 여부를 알기 위한 분석을 위해 저장되어야 한다. 이는 다수의 스크린샷들(예컨대, 20)에 걸쳐 비교될 때 캐럿 후보가 특정한 빈도로 깜박이고 있는지의 여부를 결정하는데 도움이 될 수 있다. 이 정보는 사용자가 새로운 필드 상의 마우스를 클릭하며, 탭 키를 누르는 등을 한 후에 다른 곳에서 캐럿이 다시 나타나면 그 캐럿을 식별하는데 또한 도움이 될 수도 있다. 당연히, 프로세스의 시작부분에서, 확정 큐는 일부 실시예들에서 비어 있다.

[0091] 주어진 캐럿 후보가 1135에서 캐럿 후보의 출현/소멸, 사이즈, 및 포지션에 기초하여 깜박일 것으로 확정되면, 1140에서 깜박임 캐럿에 대해 캐럿 추적 데이터가 생성된다. 이는 스크린 상의 캐럿의 포지션, 캐럿이 존재하는 그래픽 엘리먼트(즉, 액티브 엘리먼트) 등을 포함할 수도 있다. 그 다음에 검증된 후보 영역들 및 대응하는 멤버 이진 매트릭스 데이터는, 예를 들면 나중의 검증을 위해 1145에서 확정 큐에 저장된다. 도 11의 프로세스는 일부 실시예들에서 각각의 새로운 스크린샷에 대해 반복될 수도 있다.

[0092] 도 12a 내지 도 12d는 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하기 위한 프로세스(1200)를 예시하는 흐름도들이다. 프로세스(1200)에 앞서, 해상도 변경들에 대해 체크가 이루어질 수도 있고 캐럿 추적 비디오 로직이 변경들을 수용하도록 설정될 수도 있다. 예를 들어 도 9를 참조한다. 특정 실시예들에서, 프로세스(1200)는 비디오 레코더와 동시에 실행될 수도 있다. 예를 들어 도 10을 참조한다. 프로세스(1200)는 C# 및 Windows® 운영 체제를 사용하는 일 예이다. 그러나, 임의의 적합한 프로그래밍 언어, 운영 체제, 연관된 API들, 포맷들, 및 함수들이 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다.

[0093] 프로세스는 1202에서 N에 대한 BitmapData 오브젝트를 생성하기 위해 포맷 Format16bppRgb555를 사용하여 아이템 N (예컨대, 스크린샷, 스크린의 부분, 애플리케이션 윈도우 등)에 대해 LockBits를 수행하는 것으로 시작한다. LockBits는 비트맵의 직사각형 부분을 잠그고 특정된 포맷으로 화소 데이터를 판독하거나 또는 기입하는데 사용될 수 있는 임시 버퍼를 제공한다. BitmapData는 비트맵의 속성들을 저장한다.

[0094] BitmapData N 및 BitmapData N - 1(즉, 이전의 아이템에 대한 BitmapData 오브젝트)은 그 다음에 1204에서 16

개 화소들의 높이를 갖는 수평 행들로 나누어진다. 그러나, 프로세스(1200)의 이 단계 및 다른 단계들에 대한 임의의 원하는 높이(예컨대, 8 개 화소들, 32 개 화소들 등)가 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다. 동일한 수직 포지션(즉, 동일한 "행"에서 - 도 8a 내지 도 8f를 참조)에서의 BitmapData N 및 BitmapData N - 1의 각각의 수평 행에 대해, MPCMP 명령이 1206에서 실행되며, 이는 바이트 어레이들의 빠른 비교를 수행한다. MEMCMP는 행들이 동일한지의 여부의 지시를 제공한다.

[0095] 만일 모든 행들이 1208에서 BitmapData N과 BitmapData N - 1 사이에서 동일하고(즉, 적어도 하나의 대응하는 행에서 차이가 있고) 로봇에 의해 실행되고 있는 현재 활동에 기초하여 1210에서 차이가 예상되지 않았다면, 프로세스는 도 12d의 단계 1252로 그리고 다음의 스크린 캡처로 진행한다. 그러나, 차이가 1210에서 예상되었다면, 프로세스는 에러를 고치려고 시도하기 위해 도 12d의 단계 1256로 진행한다. 만일 모든 행들이 1208에서 BitmapData N과 BitmapData N - 1 사이에 동일하지 않고(즉, 적어도 하나의 대응하는 행에 차이가 있고) 1212에서 차이가 예상되지 않았으면, 프로세스는 에러를 고치려고 시도하기 위해 도 12d의 단계 1256으로 진행한다. 그러나, BitmapData N과 BitmapData N - 1 사이의 비교의 결과가 예상되는 것이 아니면(즉, 변경이 예상되지 않았을 때 예상치 못한 변경이 발생하였거나 또는 변경이 예상되었을 때 변경이 발생하지 않았으면), 일부 실시예들에서, 프로세스 단계들(1214~1250)또는 그것의 서브세트는 여전히 수행될 수도 있다는 것에 주의해야 한다.

[0096] 모든 행들이 1208에서 BitmapData N과 BitmapData N - 1 사이에서 동일하지 않고(즉, 적어도 하나의 대응하는 행에 차이가 있고) 1212에서 차이가 예상되었다면, MEMCMP 결과가 0과 동일하지 않은 동일한 행에서 BitmapData N 및 BitmapData N - 1을 위한 수평 행들이 1214에서 추출되고, 추출된 수평 행들은 그 다음에 1216에서 사이즈 16×16 화소들로 나누어진다. 예를 들어 도 8c를 참조한다. BitmapData N 및 BitmapData N - 1의 각각의 16×16 화소 블록에 대해, 블록들 사이에 차이가 있는 블록들은 1218에서 추출된다. 예를 들어 도 8f를 참조한다. 이는 Intel Intrinsics® 명령들 또는 일부 다른 적합한 능력을 루핑하는 긴 XOR 함수들의 조합을 사용하여 수행될 수도 있다.

[0097] 위와 같이, 일부 실시예들에서, 프로세싱될 수 있는 영역들의 수는 미리 결정된 수(L)로 제한된다. 특정 실시예들에서, 각각의 영역에 포함될 수 있는 블록들의 수는 제한될 수도 있다. 예를 들면, 다음의 스크린샷이 획득되기 전에 OCR이 각각의 영역 상에서 실행되는 것을 보장하기 위해 4 개의 정사각형들, 9 개의 정사각형들, 16 개의 정사각형들 등의 제한이 부과될 수도 있다. 이는 변경된 영역들의 수에 대한 제한, 주어진 변경된 영역에 포함되는 정사각형들의 수에 대한 제한, 또는 둘 다를 포함할 수 있는 "최적화 문턱값"일 수도 있다.

[0098] 각각의 영역에서 추출된 16×16 화소 블록들의 총 갯수, 영역들의 수, 또는 둘 다가 1220에서 최적화 문턱값과 비교된다. 최적화 문턱값이 1220에서 충족되면, 프로세스는 도 12d의 단계 1252로 그리고 다음 스크린 캡처로 진행한다. 문턱값이 단계 1220에서 초과되지 않으면, 근접한 16×16 화소 블록들이 일부 실시예들에서 8 개 연결되는 연결된 컴포넌트들 알고리즘일 수도 있는 연결된 컴포넌트들 알고리즘을 사용하여 1222에서 결합된다. 이는 어떤 블록들이 이웃들인지를 결정한다.

[0099] 일단 연결된 이웃들이 결정되면, 각각의 근접 블록 세트에 대한 경계 직사각형이 1224에서 결정되어, 영역을 형성한다. 이는 최고 및 최저 x 값들(즉, 가장 왼쪽 및 가장 오른쪽 블록(들))과 최고 및 최저 y 값들을 갖는 블록들(즉, 최고 및 최저 블록(들))이 포함되는 극한(extremal) 알고리즘에 의해 결정될 수도 있다. 이러한 예는 도 8f에서 볼 수 있다.

[0100] 영역에 대한 각각의 경계 직사각형에 대해, 화소 변경들은 1226에서 계산되고 이진 매트릭스에 투영된다. 32×32 영역으로 결합된 네 개의 16×16 블록들에 포함되는 글자 "E"에 대한 예시적인 이진 매트릭스(1400)가 예를 들어 도 14a에서 도시된다.

[0101] 대부분의 경우들에서, 영역은 그 안에 포함되는 멤버(들)(예컨대, 글자들, 캐럿, 화소들을 변경한 다른 형상들 등)보다 더 클 것이다. OCR 알고리즘의 속력을 증가시키기 위하여, 각각의 이진 매트릭스에 대해, 각각의 영역에 포함되는 멤버(들)는 (예컨대, 연결된 컴포넌트들 알고리즘을 사용하여) 결정되고 이진 매트릭스는 1228에서 각각의 멤버에 대해 크로핑된다. 이는 각각의 영역에서 각각의 멤버에 대해 크로핑된 매트릭스들을 생성한다. 이진 매트릭스(1400)로부터 생성된 글자 "E"에 대한 예시적인 크로핑된 멤버 매트릭스(1410)가 도 14b에 도시된다. 크로핑은 일부 실시예들에서 극한 알고리즘을 사용하여 또한 수행될 수도 있다.

[0102] 그러면 깜박이는 캐럿 영역 후보들이 1230에서 멤버 매트릭스들로부터 추출된다. 예를 들어, 후보들은 직사각형 형상을 가질 수도 있는데, 이는 일부 실시예들에서 단일 화소의 폭을 갖는 수직 라인을 잠재적으로 포함한다. 그 다음에 추출된 깜박임 캐럿 영역 후보들 및 대응하는 멤버 매트릭스 데이터는 1232에서 확정 큐

와 비교되어, 깜박임의 사이즈, 로케이션, 및 빈도를 잠재적으로 분석한다. 1234에서 깜박이면, 1236에서 깜박이는 캐럿에 대해 캐럿 추적 데이터가 생성된다. 깜박이는 캐럿 영역들 및 이들의 대응하는 멤버 매트릭스 데이터는 그 다음에 1238에서 확정 큐에 저장된다. 일부 실시예들에서, 프로세스(1200)의 이 부분은 도 11의 프로세스(1100)와 동일하거나 또는 유사할 수도 있다.

[0103] 멤버 이진 매트릭스들은 일부 실시예들에서 주어진 화소가 스크린 캡처 N - 1에서 스크린 캡처 N로 변경되었는지의 여부만을 지시한다. 따라서, 화소 데이터는 1240에서 변경된 각각의 화소에 대해 BitmapData N으로부터 리트리브(retrieve)된다. 그 다음에 1242에서 멤버 직사각형들이 생성되고 OCR을 위해 준비된다. 이는 각각의 변경된 화소를 위해 화소 데이터를 채우는 것, 캐럿 화소들을 제거하는 것, 백그라운드로 프로세싱하는 것(예컨대, 설정 변경되지 않은 화소들을 널 또는 고도로 축소된 값으로 설정하는 것) 등을 포함할 수도 있다. 캐럿 화소들이 제거되었단 경우, 캐럿 자체는 이 때 특정한 포지션, 형상, 및 이진 매트릭스 멤버들의 세트로 검출되었다고 가정될 수 있다. 이 정보는 캐럿 추적 목적으로 저장될 수 있다. 그 다음에 OCR은 1244에서 준비된 멤버 직사각형 화소 데이터에 대해 수행된다.

[0104] RPA 워크플로우에서 현재 로봇 활동에 대한 예상되는 변경은 1246에서 결정되고 검증된다. 예상되는 변경은 키 문자 큐에서의 키 누름 이벤트들, RPA 워크플로우에서 로봇 활동에 기초하여 나타나야 하는 변경들 등을 분석함으로써 결정될 수도 있다.

[0105] 검증이 1248에서 성공하였고 키 문자 큐가 사용되면, OCR 영역들에 매칭되었던 키 문자 큐 아이템들(예컨대, 키 누름 이벤트들)이 1250에서 키 문자 큐로부터 제거될 수도 있다. 문자의 다수의 인스턴스들이 존재하는 경우, 예를 들어, 키 문자 큐에서 해당 문자의 가장 오래된 인스턴스는 제거될 수도 있다. 그 다음에 UnlockBits는 1252에서 BitmapData N - 1에 대해 수행되는데, 이는 시스템 메모리로부터 이 비트맵을 잠금해제하고, BitmapData N은 1254에서 포지션 N - 1로 이동된다. 그 다음에 프로세스(1200)는 다음의 캡처된 아이템에 대한 시작으로 리턴할 수 있다.

[0106] 그러나, 검증이 1248에서 실패하였다면(예컨대, 예상되는 변경들이 발생하지 않았거나 또는 부분적으로만 발생하였다면), 시정 조치가 1256에서 시도된다. 예를 들면, 로봇은 워크플로우의 단계(들)가 실패하였음을 알 수 있으며, 얼마간의 수정 조치를 취하고, 단계(들)를 다시 시도할 수도 있다. 이는 로봇이 드라이버 레벨에서 로봇이 상호작용하려고 시도하고 있는 애플리케이션용 윈도우가 액티브 윈도우인지의 여부를 체크하는 것, 관련 있는 그래픽 엘리먼트에 대한 마우스 포인터 또는 캐럿의 로케이션을 체크하는 것, 올바른 그래픽 엘리먼트가 액티브 엘리먼트인지의 여부를 체크하는 것, 이들의 임의의 조합 등을 비제한적으로 포함할 수도 있다. 로봇은 올바른 윈도우를 액티브의, 포커싱된 윈도우로 설정하며, 원하는 액티브 엘리먼트로 포커스를 설정하며, 마우스를 이동시키고 클릭 등을 한 다음 단계(들)를 다시 시도할 수도 있다. 시정 조치를 취하는 동안, 프레임 비교 프로세스는 일부 실시예들에서 일시 중지될 수도 있다. 시정 조치가 1258에서 성공하면, 프로세스는 단계 1252로 진행할 수도 있다.

[0107] 그러나, 시정 조치가 1258에서 실패하면, 에러 로직은 1260에서 실행될 수도 있다. 에러 로직은 예외를 발생시키는 것, 로봇의 실행을 중지시키는 것 및 스크린 상에 메시지를 디스플레이하거나 또는 로봇이 원하는 액션(들)을 달성하는데 실패했음을 나타내는 메시지를 인간에게 (예컨대, 텍스트 또는 이메일을 통해) 전송하는 것을 비제한적으로 포함할 수도 있다. 특정 실시예들에서, 실패했던 액션(들)은 사용자에게 제공될 수도 있으며 그래서 사용자는 문제를 해결하려고 시도할 수 있거나, 또는 사용자는 RPA 개발자에 의해 수정될 문제를 보고할 수 있다.

[0108] 도 13a 내지 도 13c는 본 발명의 일 실시예에 따른 붙여 넣은 텍스트 추적 및 검증을 수행하기 위한 프로세스(1300)를 예시하는 흐름도들이다. 프로세스는 옵션적으로는 1305에서 키 누름 기록을 수행하는 것과 1310에서 스크린 캡처(예컨대, 비디오 기록, 스크린의 전체 또는 부분의 스크린샷들을 캡처하는 것 등)를 수행하는 것으로 시작하여, 눌러졌던 키들 및 그래픽 변경들이 발생했던 스크린 상의 로케이션(들)을 각각 결정한다. 그 다음에 로봇 액션이 1315에서 해석된다. 로봇 액션들은 마우스 클릭들, CTRL+V 누름, 우측 클릭 더하기 메뉴로부터 붙여 넣기 선택, 홈 버튼 클릭 및 애플리케이션에의 붙여 넣기 등을 비제한적으로 포함할 수도 있다. 로봇이 로케이션을 클릭하고 클립보드로부터 빠르게 붙여 넣으면, 캐럿은 누락될 수도 있고 로봇의 액션들은 상이하게 재구성될 것이 필요할 수도 있다. 클립보드로부터의 붙여 넣기가 1320에서 발생하지 않았다면, 입력된 텍스트 검출 및 캐럿 추적 로직은, 잠재적으로는 일부 실시예들에서 도 12a 내지 도 12d의 프로세스(1200)의 라인들을 따라, 1325에서 수행된다.

[0109] 그러나, 클립보드가 1320에서 로봇으로부터 최근에 붙여 넣은 (예컨대, 마지막 500 ms, 마지막 1 초 등의 내에

붙여 넣은) 데이터를 포함하면, 프레임 N과 프레임 N - 1 사이의 화소 차이들은 1330에서 계산된다. 프레임 N과 프레임 N - 1 사이에 변경들이 발생했던 예측된 동일한 포지션 영역들은 그 다음에 1335에서 추출된다. 1340에서 각각의 영역에 대해, 화소 변경들은 계산되고 그 변경들은 이진 매트릭스에 투영된다.

[0110] 1345에서 각각의 이진 매트릭스에 대해, 멤버들이, 예를 들어, 연결된 컴포넌트들 알고리즘을 사용하여 결정되고, 멤버 매트릭스들은 각각의 멤버에 대해 결정된다. 1350에서 화소 데이터는 멤버 매트릭스들에서 각각의 변경된 화소에 대해 추출되고 멤버 직사각형들이 생성된다. 각각의 멤버 직사각형은 1355에서 OCR을 위해 준비되고 OCR은 1360에서 각각의 준비된 멤버 직사각형 상에서 실행된다. OCR 결과들의 클립보드 콘텐츠에 대한 퍼지 매칭 비교가 1370에서 클립보드 모니터에 의해 제공되는 클립보드 콘텐츠를 사용하여 1365에서 각각의 OCR 결과에 대해 수행된다. 일부 실시예들에서, 클립보드 텍스트 콘텐츠는 Clipboard.GetText()를 사용하여 System.Windows.Forms.dll로부터 획득된다. 1375에서 매칭이 발견되면, (예컨대, 직사각형 형태의) 붙여 넣은 텍스트의 좌표들이 생성되고 클립보드는 1380에서 플러싱되고 프로세스는 다음 프레임을 위해 단계 1305 및 단계 1310으로 리턴한다.

[0111] 그러나, 1375에서 매칭이 발견되지 않으면(예컨대, 예상되는 붙여 넣은 콘텐츠가 나타나지 않거나 또는 부분적으로만 나타나면), 시정 조치가 1385에서 시도된다. 예를 들면, 로봇은 워크플로우의 단계(들)가 실패하였음을 알 수 있으며, 얼마간의 수정 조치를 취하고, 단계(들)를 다시 시도할 수도 있다. 이는 로봇이 드라이버 레벨에서 로봇이 상호작용하려고 시도하고 있는 애플리케이션용 윈도우가 액티브 윈도우인지의 여부를 체크하는 것, 관련 있는 그래픽 엘리먼트에 대한 마우스 포인터 또는 캐럿의 로케이션을 체크하는 것, 올바른 그래픽 엘리먼트가 액티브 엘리먼트인지의 여부를 체크하는 것, 이들의 임의의 조합 등을 비제한적으로 포함할 수도 있다. 로봇은 올바른 윈도우를 액티브의, 포커싱된 윈도우로 설정하며, 원하는 액티브 엘리먼트로 포커스를 설정하며, 마우스를 이동시키고 클릭 등을 한 다음 단계(들)를 다시 시도할 수도 있다. 시정 조치를 취하는 동안, 프레임 비교 프로세스는 일부 실시예들에서 일시 중지될 수도 있다. 시정 조치가 1390에서 성공하면, 프로세스는 단계 1380으로 진행할 수도 있다.

[0112] 그러나, 시정 조치가 1390에서 실패하면, 에러 로직은 1395에서 실행될 수도 있다. 에러 로직은 예외를 발생시키는 것, 로봇의 실행을 중지시키는 것 및 스크린 상에 메시지를 디스플레이하거나 또는 로봇이 원하는 액션(들)을 달성하는데 실패했음을 나타내는 메시지를 인간에게 (예컨대, 텍스트 또는 이메일을 통해) 전송하는 것을 비제한적으로 포함할 수도 있다. 특정 실시예들에서, 실패했던 액션(들)은 사용자에게 제공될 수도 있으며 그래서 사용자는 문제를 해결하려고 시도할 수 있거나, 또는 사용자는 RPA 개발자에 의해 수정될 문제를 보고할 수 있다.

[0113] 도 15는 본 발명의 일 실시예에 따른 CV와 로봇 활동들의 예상되는 결과들을 사용하여 액티브 엘리먼트들을 결정하고 로봇 액션들을 검증하기 위한 프로세스를 예시하는 흐름도이다. 그 프로세스는 1510에서 프레임이 미리 결정된 문턱값을 초과하는 만큼 이전 프레임으로부터 변경되었는지의 여부를 결정하는 것으로 시작한다. 이는 프레임의 특정한 부분을 초과하여 (예컨대, 2%를 초과하여) 변경되었는지의 여부, 화소들의 미리 결정된 수를 초과하여(예컨대, 200을 초과하여) 변경되었는지의 여부), 프레임에서 텍스트 엔트리를 허가하는 그래픽 엘리먼트들의 로케이션들 밖에서 변경들이 발생하는지의 여부 등을 체크하는 것을 포함할 수도 있다.

[0114] 문턱값이 1520에서 초과된다면, 스크린 상의 그래픽 엘리먼트들의 적어도 일부가 또한 변경되었을 가능성이 있다. CV 프리프로세싱이 메모리에 저장될 수도 있는 그래픽 엘리먼트 유형들 및 로케이션들을 식별하기 위해 1530에서 수행된다. 프레임 비교 프로세스는 일부 실시예들에서 CV 프리프로세싱이 수행되는 동안 일시 중지될 수도 있다.

[0115] 문턱값이 1520에서 초과되지 않았다면, 또는 CV 프리프로세싱이 1530에서 완료된 후, 이전 프레임과 현재 프레임 사이에 추가된 키 문자 큐에서 새로 나타난 엘리먼트들은 1540에서 결정된다. 키 문자 큐에 새로 나타난 엘리먼트들이 있다면, 이것들은 스크린 상의 적합한 그래픽 엘리먼트로 나타났다고 가정될 수도 있다. 현재 프레임에서 스크린이 변경된 로케이션(들)은 그 다음에 1550에서 결정되고 1560에서 그래픽 엘리먼트들의 로케이션들에 대한 변경들을 매칭시키려는 시도가 이루어진다. 그래픽 엘리먼트들 중 하나의 그래픽 엘리먼트 내에서만 변경들이 발생했다면, 매칭된 그래픽 엘리먼트는 1570에서 액티브 엘리먼트로서 설정된다. 그러나, 만일 다수의 그래픽 엘리먼트들 내에서 변경들이 발생했거나 또는 그래픽 엘리먼트에서 변경이 발견되지 않았다면, 시정 조치(예컨대, 콘텐츠가 나타나지 않았던 필드로부터 콘텐츠를 제거하는 것과 올바른 필드에 콘텐츠를 삽입하려고 시도하는 것)가 1580에서 수행된다. 다음 프레임은 그 다음에 1590에서 페치(fetch)되고 프로세스는 반복된다.

[0116] 도 16은 본 발명의 일 실시예에 따른, RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하도록 구성되는 시스템(1600)을 예시하는 아키텍처 도면이다. 시스템(1600)은 사용자 컴퓨팅 시스템들, 이를테면 데스크톱 컴퓨터(1602), 태블릿(1604), 및 스마트 폰(1606)을 포함한다. 그러나, 임의의 원하는 컴퓨팅 시스템이 스마트 워치들, 랩톱 컴퓨터들, 사물 인터넷(Internet-of-Things; IoT) 디바이스들, 차량 컴퓨팅 시스템들 등을 비제한적으로 포함하여 본 발명의 범위로부터 벗어나는 일 없이 사용될 수도 있다.

[0117] 각각의 컴퓨팅 시스템(1602, 1604, 1606)은 자신의 동작 또는 별도의 로봇의 동작에 관계된 스크린샷들, 키스트로크들, 실행중인 애플리케이션들, 애플리케이션 시각적 엘리먼트들, 시각적 엘리먼트 로케이션들, 애플리케이션 로케이션들, 실행되고 있는 로봇 워크플로우 및 현재 단계(들) 등을 기록하는 자신 상에서 실행하고 있는 디지털 프로세스(1610)를 갖는다. 실제로, 스크린 그래픽스, 로봇 입력들, 디스플레이 엘리먼트들 등에 관계된 임의의 원하는 정보는 본 발명의 범위로부터 벗어나는 일 없이 기록될 수도 있다. 특정 실시예들에서, 비디오 및 키스트로크 기록들만이 처음에 캡처되고, 다른 정보는CV를 사용하여 그 뒤에 결정된다. 그러나, 추가적인 정보는 CV 프로세스에 포커싱하고 그 프로세스를 개선하는데 도움이 될 수도 있다. 디지털 프로세스들(1610)은 RPA 설계자 애플리케이션, 운영 체제의 일부, 개인용 컴퓨터(PC) 또는 스마트 폰을 위한 다운로드가능 애플리케이션을 통해 생성되는 로봇들일 수도 있거나, 또는 본 발명의 범위로부터 벗어나지 않는 임의의 다른 소프트웨어 및/또는 하드웨어일 수도 있다. 실제로, 일부 실시예들에서, 하나 이상의 디지털 프로세스(1610)의 로직은 물리적 하드웨어를 통해 부분적으로 또는 완전히 구현된다.

[0118] 디지털 프로세스들(1610)은 기록된 스크린샷들, 키스트로크들, 실행중인 애플리케이션들, 애플리케이션 엘리먼트들 및 로케이션들, 이들의 조합 등을 네트워크(1620)(예컨대, 로컬 영역 네트워크(local area network; LAN), 모바일 통신 네트워크, 위성 통신 네트워크, 인터넷, 이들의 임의의 조합 등)를 통해 서버(1630)에 전송한다. 일부 실시예들에서, 서버(1630)는 수행자 애플리케이션을 실행할 수도 있고 데이터는 심박 메시지의 일부로서 주기적으로 전송될 수도 있다. 서버(1630)는 디지털 프로세스들(1610)로부터의 정보를 데이터베이스(1640)에 저장한다.

[0119] 서버(1630)는 서버(1630)가 데이터를 수신하는 컴퓨팅 시스템들(즉, 컴퓨팅 시스템들(1602, 1604, 1606))을 위한 스크린 응답 검증 로직의 인스턴스들(1632)을 실행한다. 서버는 인스턴스들(1632)로부터의 결과들을 분석하여 로봇들이 그들의 컴퓨팅 시스템들과 상호작용할 때 무엇을 하고 있는지를 결정할 수도 있다. 워크플로우에서의 단계들은 인스턴스들(1632)이 로봇이 하려고 시도하는 일을 결정할 수 있는 것을 보장하기 위해 비디오 및 키 누름들에 매핑되고 동기화될 수도 있다. 검증이 성공했는지 또는 실패했는지, 실패가 발생했다면 그 실패의 성질, 로봇이 위치한 단계 등에 관한 정보는 디지털 프로세스들(1610)에 의해 모니터링되는 로봇들의 실행을 검증하기 위해 각각의 사용자 컴퓨팅 시스템들(1602, 1604, 1606)에 다시 전송될 수도 있다. 일부 실시예들에서, 서버(1630)에 의한 검증은 실시간으로 또는 거의 실시간으로 발생할 수도 있고 그래서 프로세스들(1610)은 RPA 로봇이 각각의 컴퓨팅 시스템 상에서 실행할 때 검증을 수행할 수 있다.

[0120] 도 17은 본 발명의 일 실시예에 따른 RPA를 위한 로봇 실행의 스크린 응답 검증을 수행하기 위한 프로세스(1700)를 예시하는 흐름도이다. 그 프로세스는 1705에서 사용자 인터페이스에서 그래픽 엘리먼트들 및 연관된 경계 직사각형들을 결정하기 위해 CV 알고리즘을 적용하는 것으로 시작한다. 결정된 그래픽 엘리먼트들은 일부 실시예들에서 RPA 활동을 위한 타겟 그래픽 엘리먼트를 포함할 수도 있다. 1710에서 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 존재하는지의 여부의 결정이 이루어진다. 현재 스크린샷 프레임과 이전 스크린샷 프레임은 전체 사용자 인터페이스 또는 상기 사용자 인터페이스의 일부를 포함할 수도 있다. 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에 차이가 1715에서 존재하고 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이의 차이 RPA 로봇의 워크플로우에서의 활동에 기초하여 1720에서 예상될 때, 하나 이상의 변경이 1725에서 RPA 로봇의 워크플로우의 활동에 기초하여 하나 이상의 예상된 스크린 변경에 대해 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이에서 검증된다. 일부 실시예들에서, 검증은 현재 스크린샷 프레임이 이전 스크린샷 프레임과 상이한 하나 이상의 로케이션 상에서 OCR을 수행하는 것과 키 문자 큐에서의 문자들에 OCR에 의해 인식된 문자들을 매칭시키는 것을 포함한다.

[0121] 검증이 1730에서 성공할 때, 프로세스는 다음 프레임들로 진행하고(즉, 다음 프레임을 현재 스크린샷 프레임으로 하고 현재 스크린샷 프레임을 이전 스크린샷 프레임으로 하고) 워크플로우에서의 다음 활동 또는 동일한 활동에 의해 영향을 받을 다음 변경으로 진행한다. 그러나, 검증이 실패했을 때, 1740에서 예외를 발생시키거나 또는 시정 조치가 취해진다. 만일 시정 조치가 1740에서 취해지고 그것이 성공적이면, 프로세스는 단계 1740으로 진행할 수도 있다.

[0122] 단계 1710으로 리턴하여, 만일 차이가 1715에서 발생하지 않고 1745에서 차이가 예상되지 않으면, 프로세스는 단계 1735로 진행한다. 그러나, 차이가 1745에서 예상되었다면, 프로세스는 단계 1740으로 진행한다.

[0123] 일부 실시예들에서, 시정 조치는 로봇이 상호작용하려고 시도하고 있는 애플리케이션용 윈도우가 액티브 윈도우인지의 여부를 RPA 로봇이 드라이버 레벨에서 체크하는 것, 변경이 발생할 것으로 예상되었던 그래픽 엘리먼트에 대한 마우스 포인터 또는 캐럿의 로케이션을 체크하는 것, 올바른 그래픽 엘리먼트가 액티브 엘리먼트인지의 여부를 체크하는 것, 또는 이들의 임의의 조합을 포함한다. 특정 실시예들에서, 시정 조치는 RPA 로봇이 올바른 윈도우를 액티브, 포커싱된 윈도우로서 설정하는 것, 포커스를 액티브 엘리먼트에 설정하는 것, 마우스를 이동시키고 마우스 클릭 이벤트를 유발하는 것, 또는 이들의 임의의 조합을 포함한다. 일부 실시예들에서, 예외를 발생시키는 것은 RPA 로봇의 동작을 중지시키는 것, 스크린 상에 메시지를 디스플레이하는 것, RPA 로봇이 하나 이상의 예상된 스크린 변경을 달성하는데 실패했음을 나타내는 메시지를 전송하는 것, 실패한 하나 이상의 액션을 열거하는 것, 또는 이들의 조합을 포함한다.

[0124] 일부 실시예들에서, 시간 윈도우 동안 발생했던 RPA 로봇에 의해 생성된 키 누름 이벤트들을 포함하는 키 문자 큐가 1750에서 생성된다. 특정 실시예들에서, 매칭되는 문자들은 1725에서 검증 동안 키 문자 큐로부터 제거된다. 일부 실시예들에서, 키 문자 큐는 누름 이벤트의 문자와 시간 윈도우 동안 키 누름 이벤트들 각각에 대해 키 누름 이벤트가 발생했던 시간을 포함하는 FIFO 큐이다.

[0125] 일부 실시예들에서, 검증은 현재 스크린샷 프레임이 이전 스크린샷 프레임과 상이한 하나 이상의 영역을 결정하는 것, 결정된 하나 이상의 영역에서 하나 이상의 연결된 멤버를 추출하는 것, 및 추출된 하나 이상의 연결된 멤버에 대해 OCR을 수행하는 것, 하나 이상의 인식된 문자, 하나 이상의 인식된 문자 시퀀스, 또는 이들 둘 다, 및 각각의 포지션들을 생성하는 것을 포함한다. 검증은 하나 이상의 인식된 문자, 하나 이상의 인식된 문자 시퀀스, 또는 이들 둘 다를, 키 문자 큐와 또는 상기 활동에 기초하여 나타날 것으로 예상되는 하나 이상의 문자, 하나 이상의 문자 시퀀스, 또는 이들 둘 다와 비교하는 것을 포함할 수도 있다. 특정 실시예들에서, 타겟 그래픽 엘리먼트와 타겟 그래픽 엘리먼트에 입력될 콘텐츠는 RPA 로봇의 워크플로우에서의 활동에 기초하여 결정되고, 검증은 현재 스크린샷 프레임과 이전 스크린샷 프레임 사이의 하나 이상의 변경이 타겟 그래픽 엘리먼트의 로케이션과 매칭되는지의 여부와 하나 이상의 변경이 타겟 그래픽 엘리먼트에 입력될 콘텐츠와 매칭되는지의 여부를 검증하는 것을 포함한다. 일부 실시예들에서, 활동은 그래픽 엘리먼트에 텍스트를 입력하는 것, 그래픽 엘리먼트로 캐럿을 이동시키는 것, 그래픽 엘리먼트에 콘텐츠를 붙여 넣는 것, 또는 이들의 조합을 포함한다. 특정 실시예들에서, RPA 로봇의 워크플로우의 활동에 기초한 하나 이상의 예상된 스크린 변경은 양식의 제출, 새로운 애플리케이션 윈도우의 열기, 또는 현재 애플리케이션 윈도우의 변경을 포함한다.

[0126] 도 6a, 도 6b, 도 7, 도 9 내지 도 13b, 도 15, 및 도 17에서 수행되는 프로세스 단계들은 본 발명의 실시예들에 따라, 도 6a, 도 6b, 도 7, 도 9 내지 도 13b, 도 15, 및 도 17에서 설명된 프로세스(들) 중 적어도 일부를 수행하기 위한 프로세서(들)를 위한 명령들을 인코딩하는 컴퓨터 프로그램에 의해 수행될 수도 있다. 컴퓨터 프로그램은 비일시적 컴퓨터 판독가능 매체 상에 구현될 수도 있다. 컴퓨터 판독가능 매체는 데이터를 저장하는데 사용되는 하드 디스크 드라이브, 플래시 디바이스, RAM, 테이프, 및/또는 임의의 다른 이러한 매체 또는 매체들의 조합일 수도 있지만 이것들로 제한되지 않는다. 컴퓨터 프로그램은 컴퓨터 판독가능 매체 상에 또한 저장될 수도 있는 도 6a, 도 6b, 도 7, 도 9 내지 도 13b, 도 15, 및 도 17에서 설명되는 프로세스 단계들의 전체 또는 부분을 구현하기 위해 컴퓨팅 시스템의 프로세서(들)(예컨대, 도 5의 컴퓨팅 시스템(500)의 프로세서(들)(510))를 제어하기 위한 인코딩된 명령들을 포함할 수도 있다.

[0127] 컴퓨터 프로그램은 하드웨어, 소프트웨어, 또는 하이브리드 구현예로 구현될 수 있다. 컴퓨터 프로그램은 서로 동작적으로 통신하는, 그리고 정보 또는 명령들을 디스플레이하기 위해 전달하도록 설계되는 모듈들로 구성될 수 있다. 컴퓨터 프로그램은 범용 컴퓨터, ASIC, 또는 임의의 다른 적합한 디바이스 상에서 동작하도록 구성될 수 있다.

[0128] 본 발명의 다양한 실시예들의 컴포넌트들은, 본 개시에서 일반적으로 설명되고 도면들에서 예시된 바와 같이, 매우 다양한 상이한 구성들로 배열되고 설계될 수도 있다는 것이 쉽사리 이해될 것이다. 따라서, 본 발명의 실시예들의 상세한 설명은, 첨부된 도면들에서 나타낸 바와 같이, 청구된 바와 같은 발명의 범위로 제한될 의도는 아니고, 본 발명의 선택된 실시예들을 나타낼 뿐이다.

[0129] 이 명세서의 전체에 걸쳐 설명되는 본 발명의 특징들, 구조들, 또는 특성들은 하나 이상의 실시예에서 임의의 적합한 방식으로 결합될 수도 있다. 예를 들어, "특정한 실시예들", "일부 실시예들", 또는 유사한 언어표현에 대한 이 명세서의 전체에 걸친 언급은 실시예에 관련하여 설명된 특정 특징, 구조, 또는 특성이 본 발명의 적어

도 하나의 실시예에 포함되는 것을 의미한다. 따라서, 이 명세서의 전체에 걸친 "특정 실시예들에서", "일부 실시예에서", "다른 실시예들에서", 또는 유사한 언어표현의 문구들의 출현은 모두가 반드시 실시예들의 동일한 그룹을 지칭하는 것은 아니고 설명된 특징들, 구조들, 또는 특성들은 하나 이상의 실시예에서 임의의 적합한 방식으로 결합될 수도 있다.

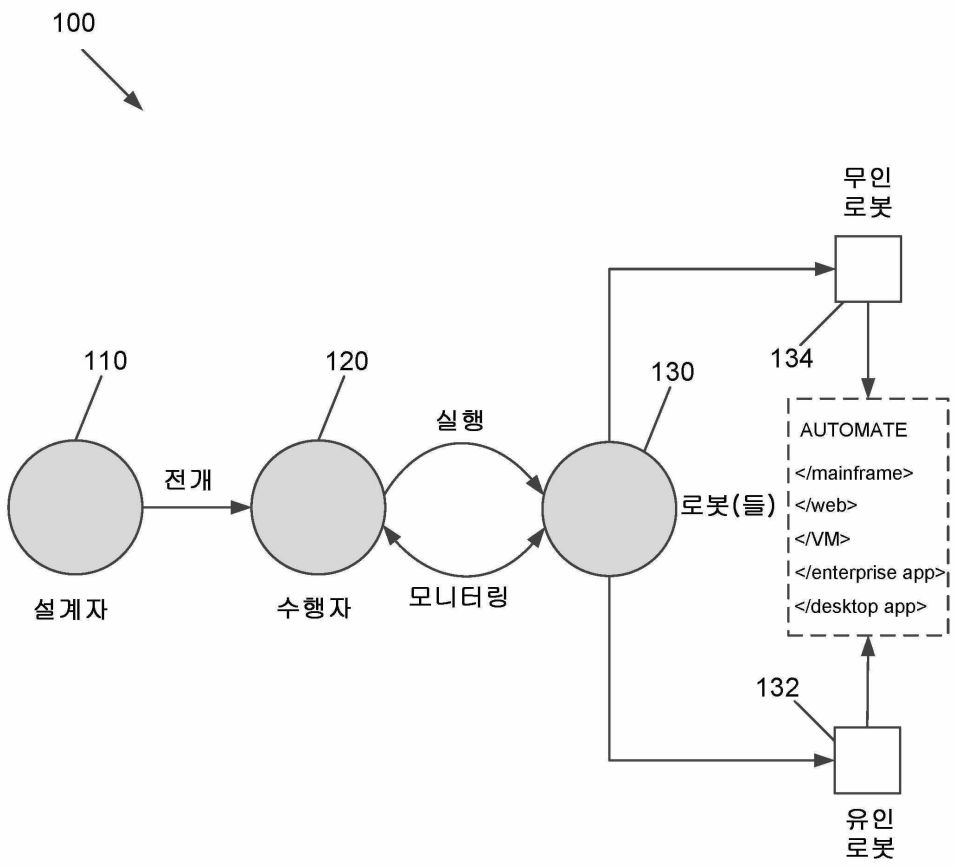
[0130] 특징들, 장점들, 또는 유사한 언어에 대한 이 명세서의 전체에 걸친 언급은 본 발명으로 실현될 수도 있는 특징들 및 장점들의 모두가 본 발명의 임의의 단일 실시예이거나 또는 그러한 실시예 안에 있음을 의미하지 않는다는 것에 주의해야 한다. 오히려, 특징들 및 장점들을 언급하는 언어표현은 일 실시예에 관련하여 설명되는 특정 특징, 장점, 또는 특성이 본 발명의 적어도 하나의 실시예에 포함된다는 것을 의미하는 것으로 이해된다. 따라서, 이 명세서의 전체에 걸친 특징들 및 장점들, 및 유사한 언어의 논의는 동일한 실시예를 참조하지만 반드시 그런 것은 아닐 수도 있다.

[0131] 더욱이, 본 발명의 설명된 특징들, 장점들, 및 특성들은 하나 이상의 실시예에서 임의의 적합한 방식으로 결합될 수도 있다. 본 기술분야의 통상의 기술자는 본 발명이 특정 실시예의 특정 특징들 또는 장점들 하나 이상 없이도 실시될 수 있다는 것을 인식할 것이다. 다른 경우들에서, 추가적인 특징들 및 장점들은 본 발명의 모든 실시예들에서 존재하지 않을 수도 있는 특정 실시예들에서 인식될 수도 있다.

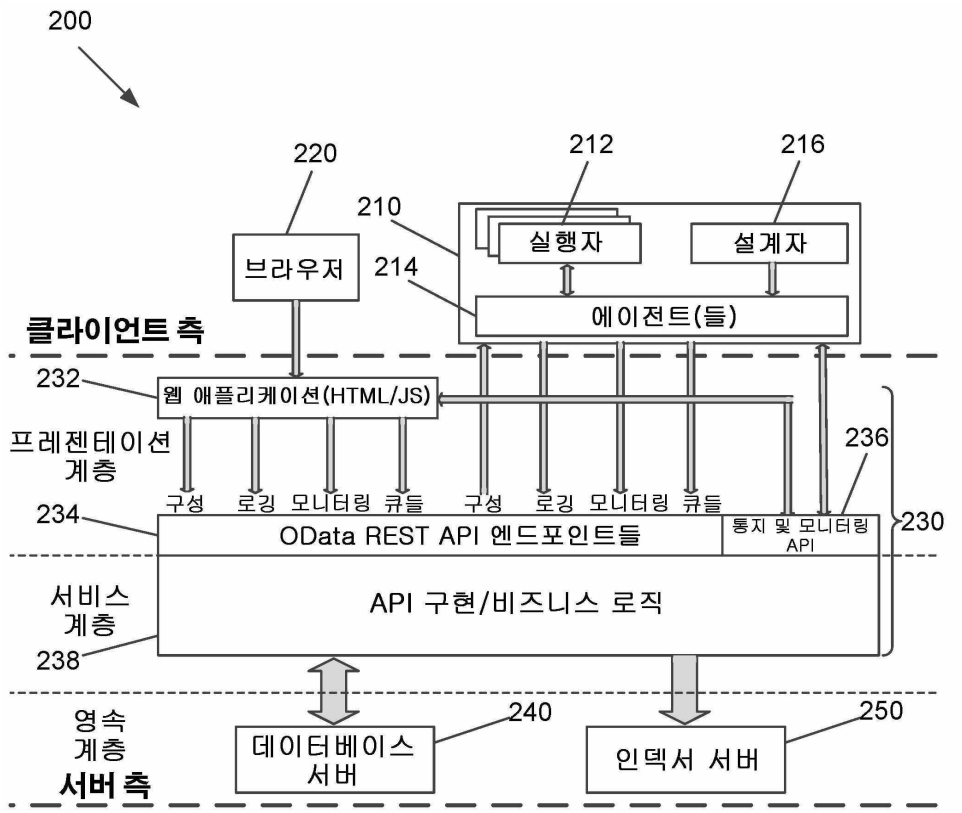
[0132] 본 기술분야의 통상의 기술을 가진 자는 위에서 논의된 바와 같은 본 발명이 상이한 순서의 단계들로, 그리고/또는 개시된 것들과는 상이한 구성들의 하드웨어 엘리먼트들로 실시될 수도 있다는 것을 쉽사리 이해할 것이다. 그러므로, 본 발명이 이들 바람직한 실시예들에 기초하여 설명되었지만, 특정한 수정들, 변경들, 및 대안적 구성들이 본 발명의 정신 및 범위 내에 있으면서도 명백할 것이라는 것이 본 기술분야의 통상의 기술자에게 명백할 것이다. 그러므로, 본 발명의 한계와 경계를 결정하기 위하여, 첨부의 청구항들이 참조되어야 한다.

도면

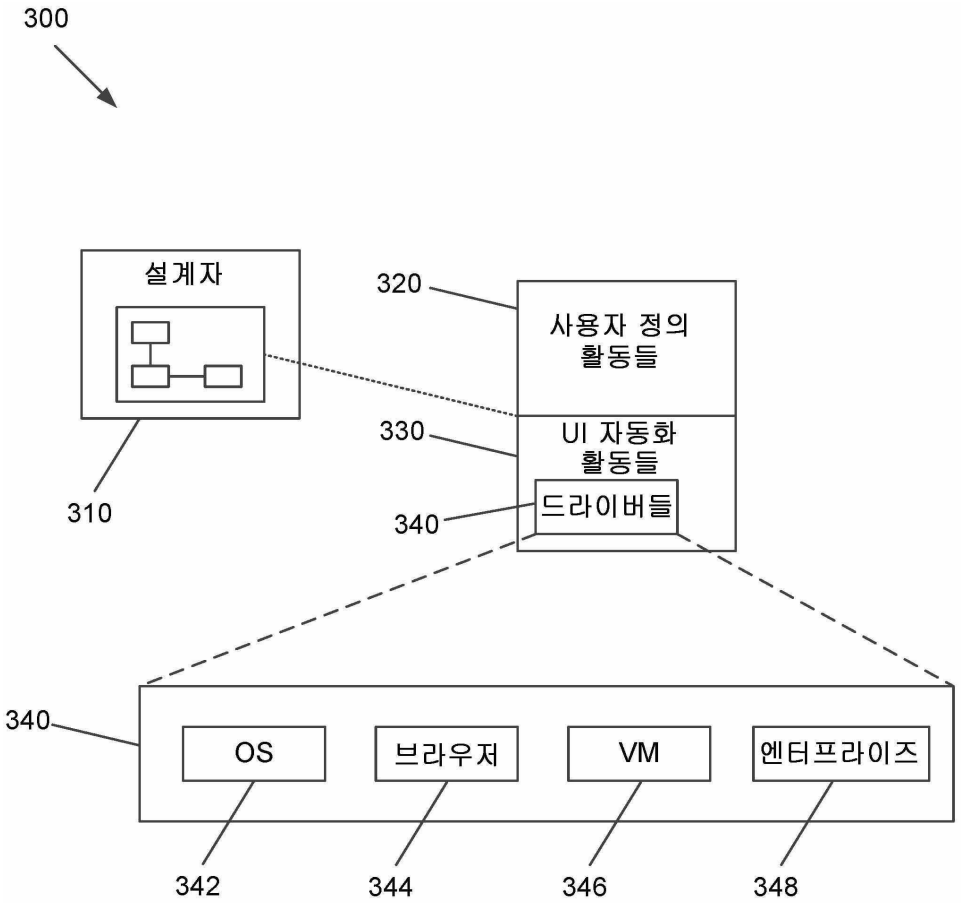
도면1



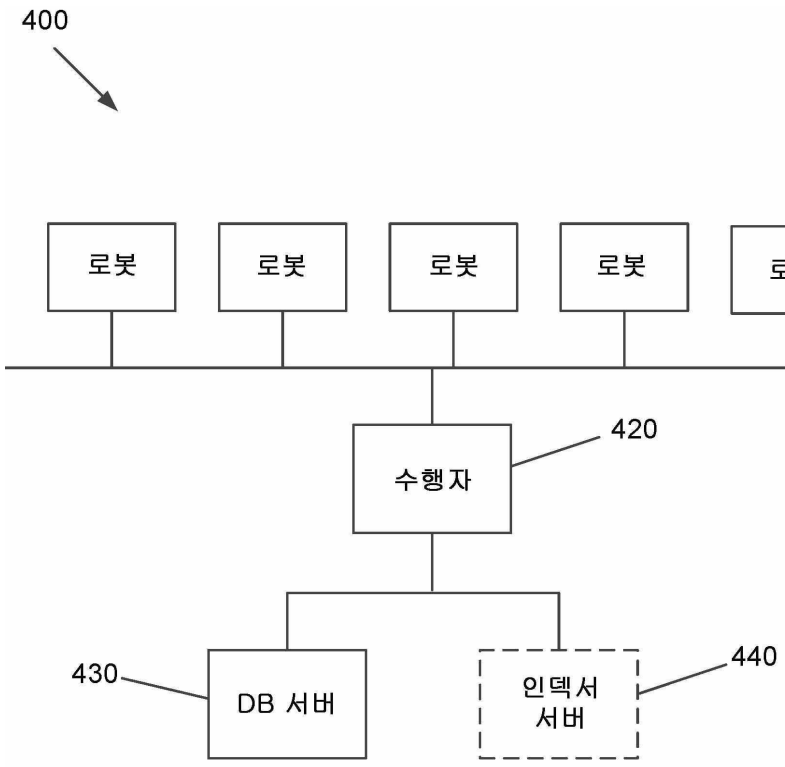
도면2


200
220
212
216
210
실행자
설계자
브라우저
214
에이전트(들)
클라이언트 측
232
웹 애플리케이션(HTML/JS)
프레젠테이션
계층
236
구성
로깅
모니터링
큐들
구성
로깅
모니터링
큐들
234
OData REST API 엔드포인트들
통지 및 모니터링
API
230
서비스
계층
API 구현/비즈니스 로직
238
영속
계층
240
250
데이터베이스
서버
인덱서 서버
서버 측

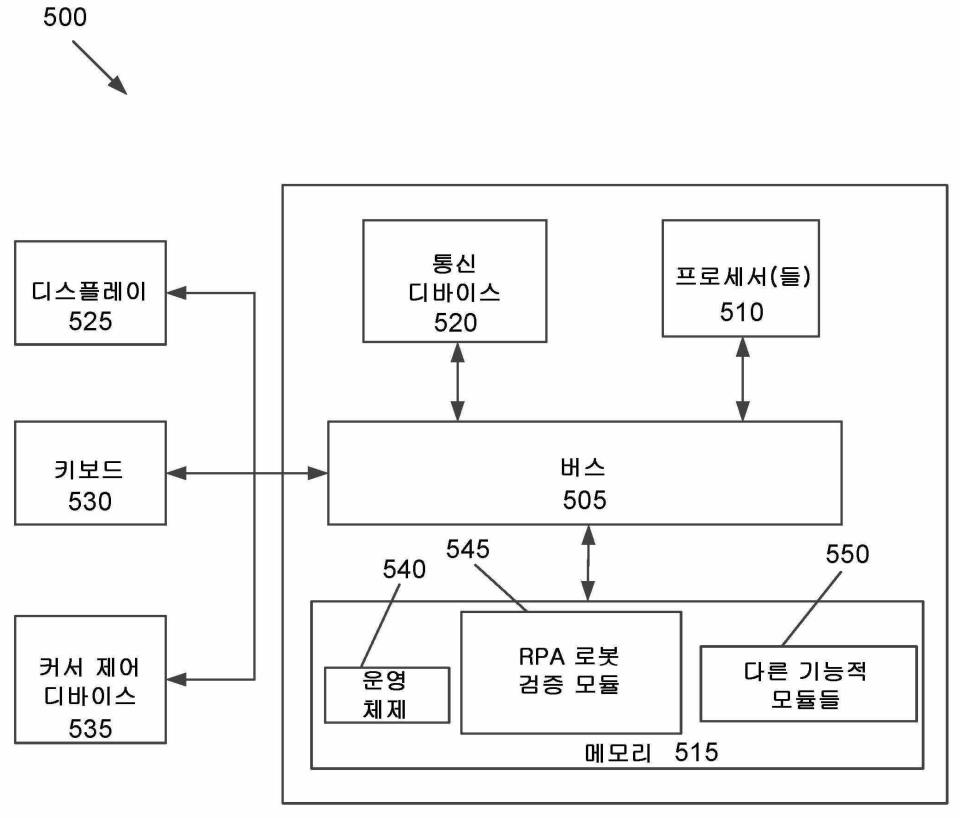
도면3


300
설계자
310
320
사용자 정의
활동들
330
UI 자동화
활동들
340
드라이버들
340
OS
브라우저
VM
엔터프라이즈
342
344
346
348

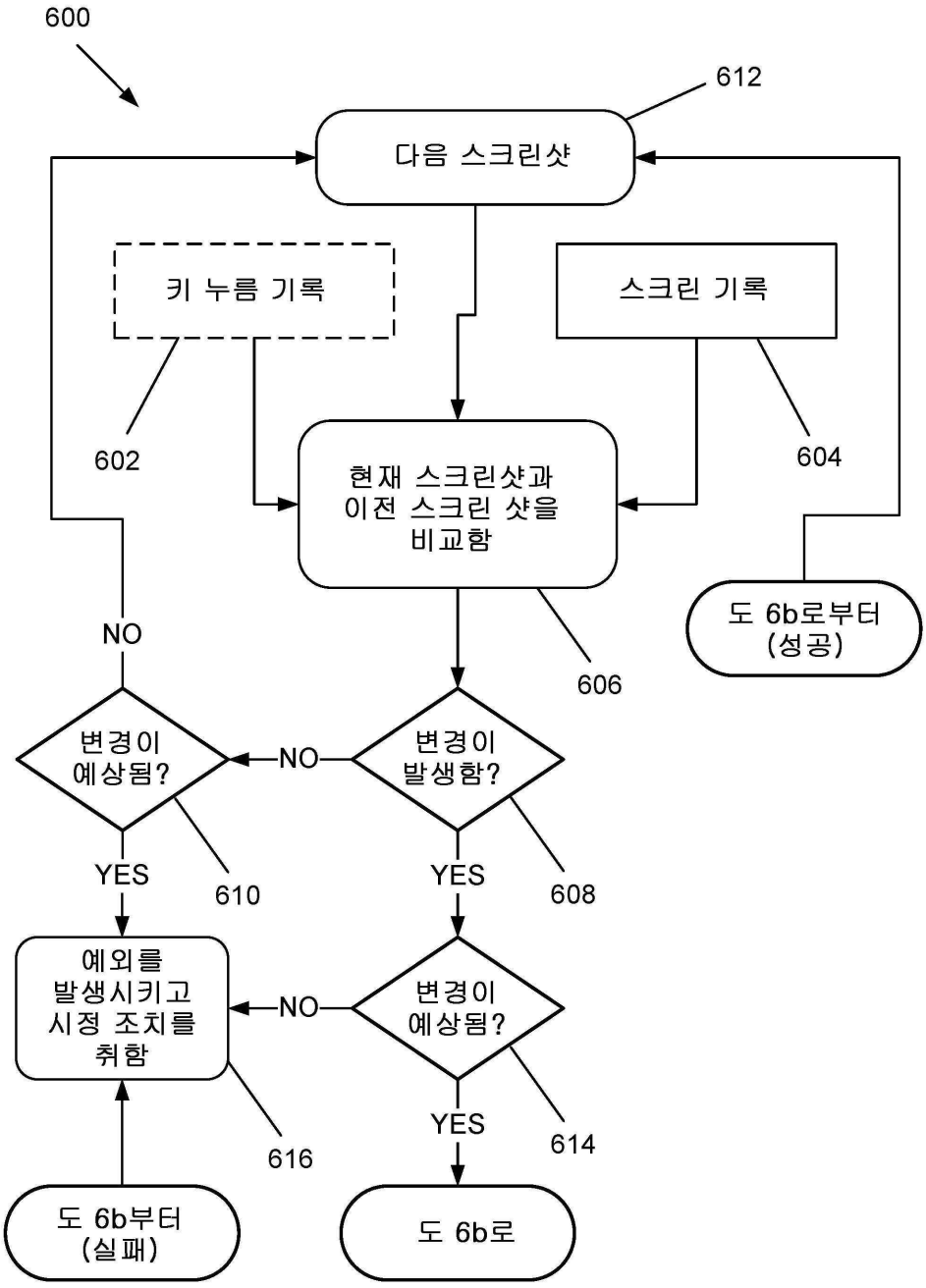
도면4


400
410
로봇
로봇
로봇
로봇
로봇
420
수행자
430
DB 서버
440
인덱서
서버

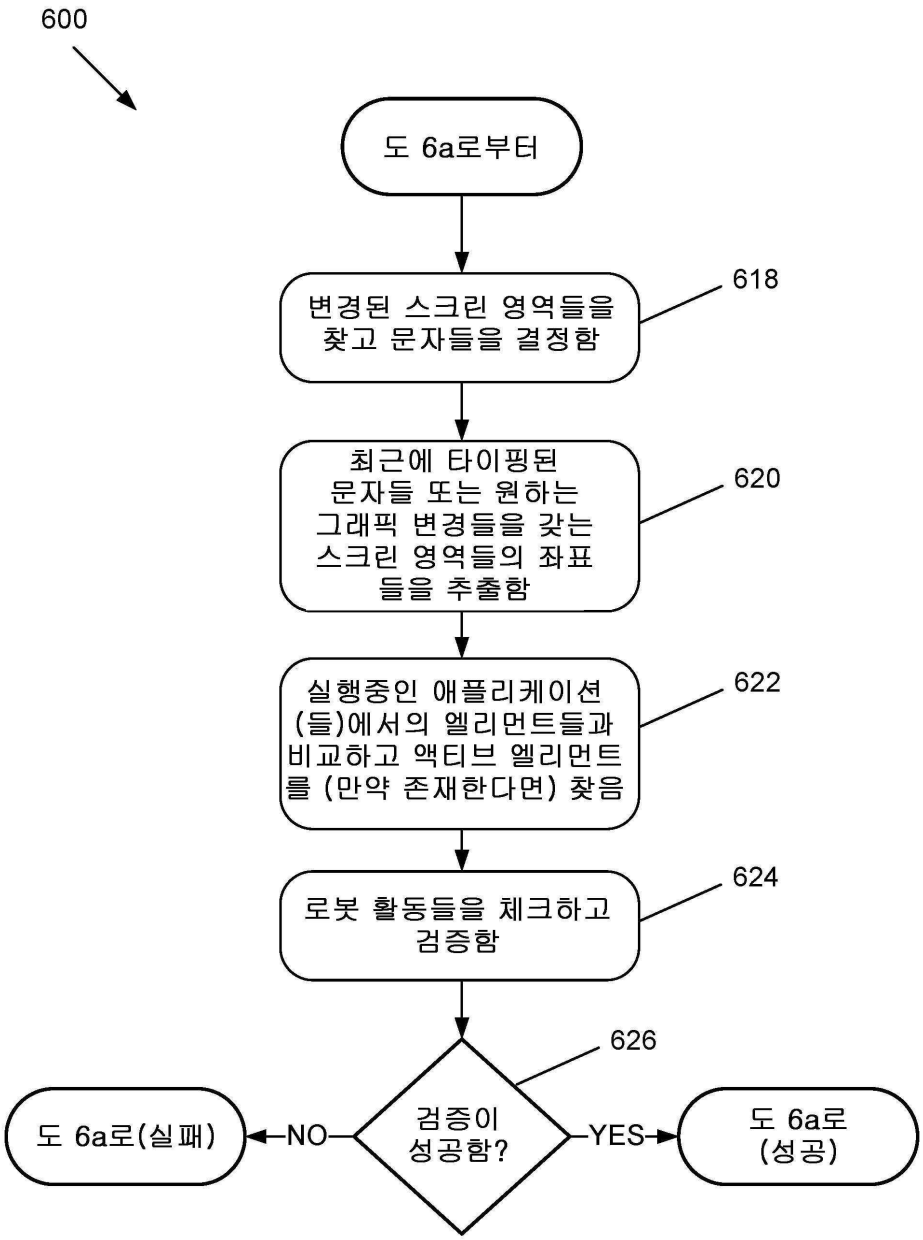
도면5


500
디스플레이
525
키보드
530
커서 제어
디바이스
535
통신
디바이스
520
프로세서(들)
510
버스
505
545
540
550
운영
체제
RPA 로봇
검증 모듈
다른 기능적
모듈들
메모리 515

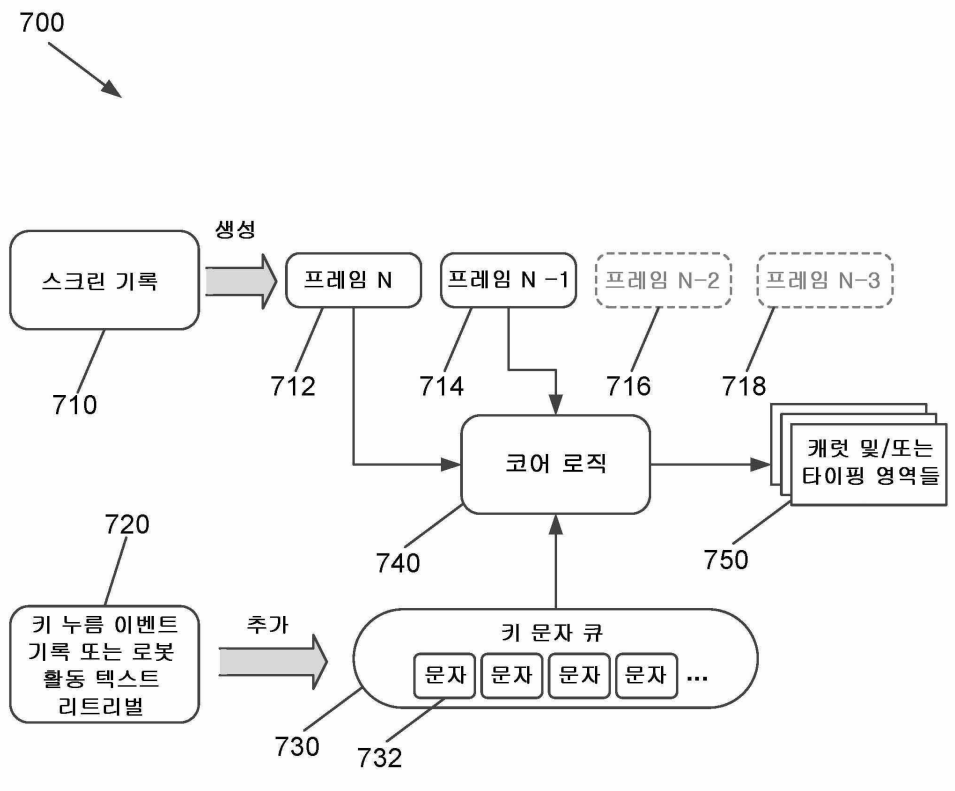
도면6a


600
612
다음 스크린샷
키 누름 기록
스크린 기록
602
604
현재 스크린샷과
이전 스크린 샷을
비교함
606
도 6b로부터
(성공)
NO
변경이
예상됨?
NO
변경이
발생함?
610
608
YES
YES
예외를
발생시키고
시정 조치를
취함
NO
변경이
예상됨?
616
614
YES
도 6b부터
(실패)
도 6b로

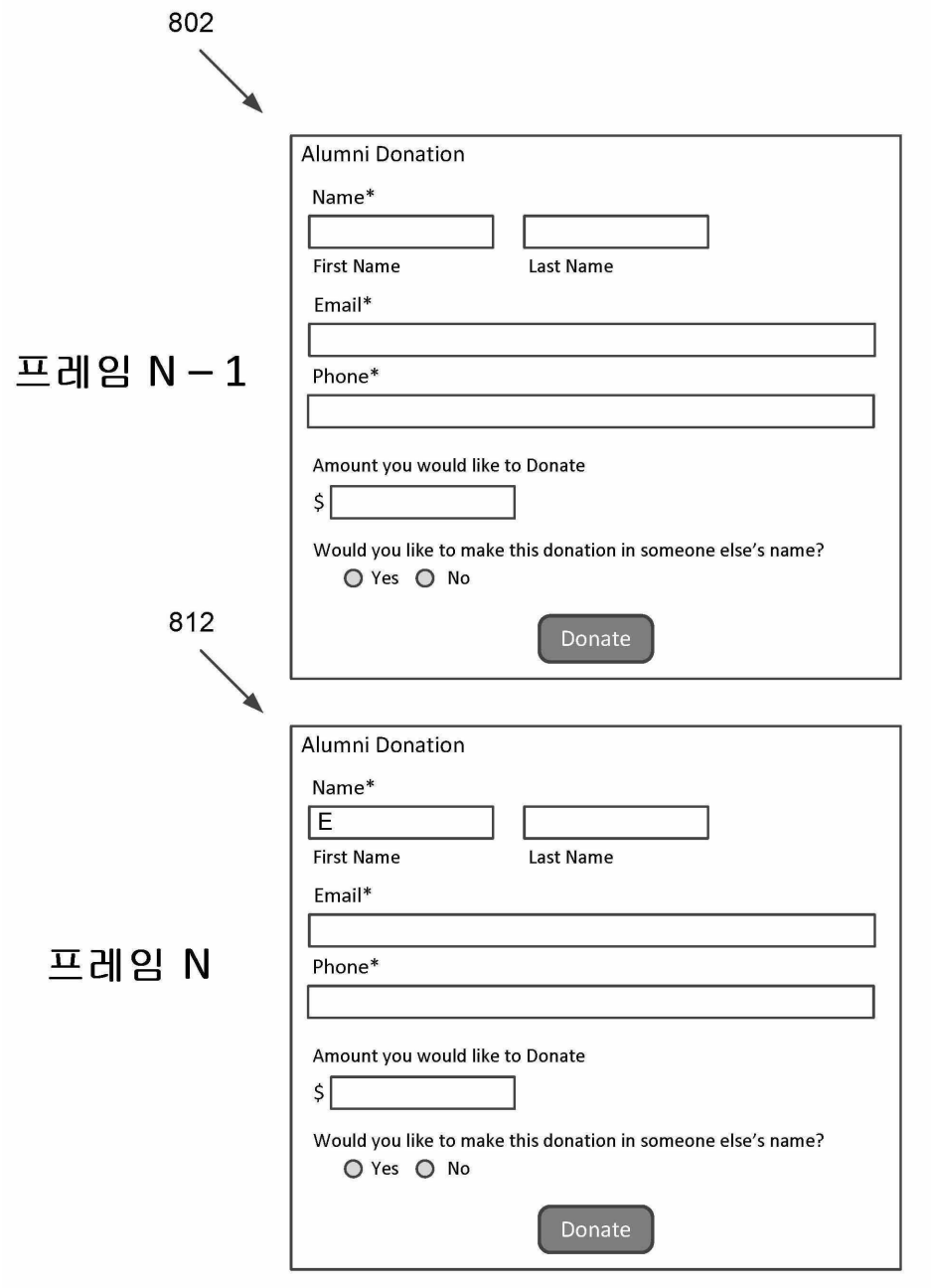
도면6b


600
도 6a로부터
변경된 스크린 영역들을 찾고 문자들을 결정함
618
최근에 타이핑된 문자들 또는 원하는 그래픽 변경들을 갖는 스크린 영역들의 좌표들을 추출함
620
실행중인 애플리케이션(들)에서의 엘리먼트들과 비교하고 액티브 엘리먼트를 (만약 존재한다면) 찾음
622
로봇 활동들을 체크하고 검증함
624
검증이 성공함?
626
NO
YES
도 6a로(실패)
도 6a로(성공)

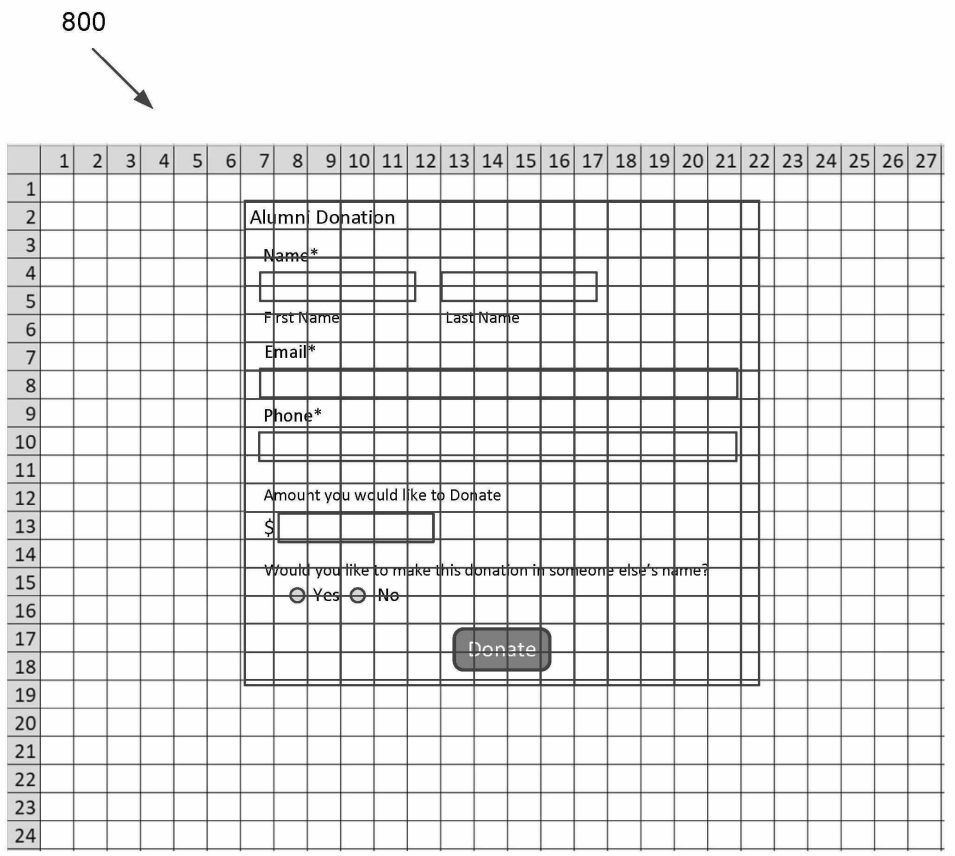
도면7


700
생성
스크린 기록
프레임 N
프레임 N -1
프레임 N-2
프레임 N-3
710
712
714
716
718
코어 로직
캐럿 및/또는
타이핑 영역들
720
740
750
키 누름 이벤트
기록 또는 로봇
활동 텍스트
리트리벌
추가
키 문자 큐
문자
문자
문자
문자
...
730
732

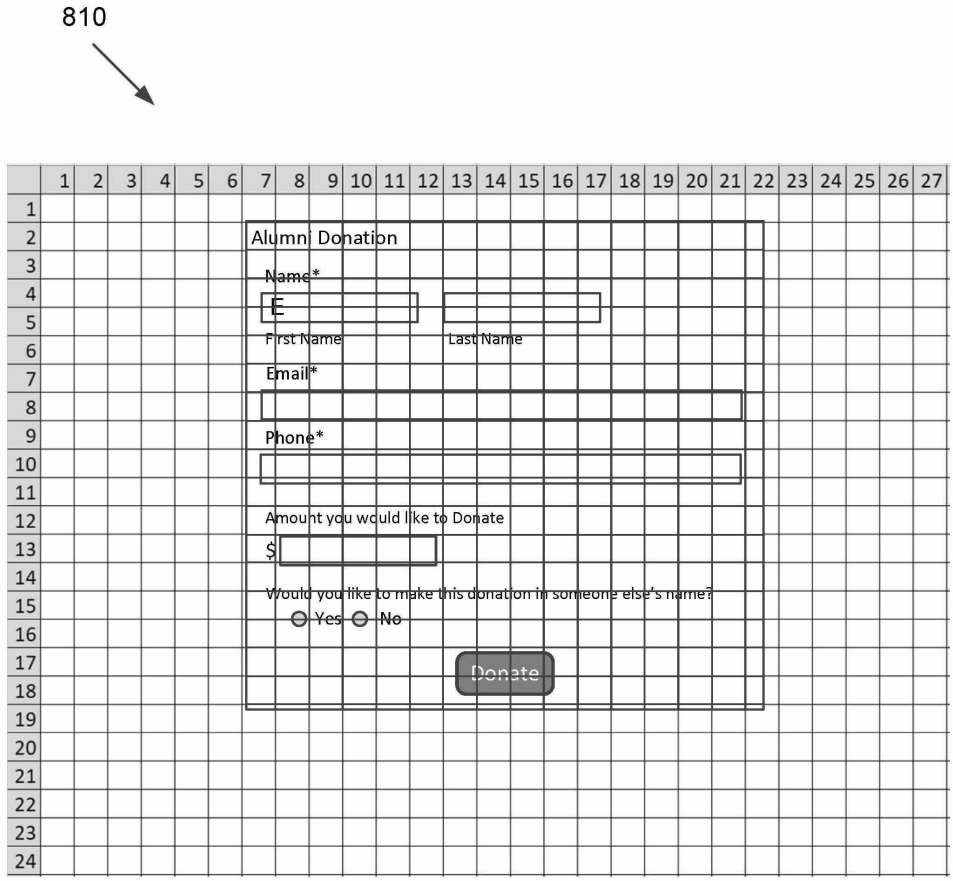
도면8a


802
프레임 N – 1
Alumni Donation
Name*
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate
812
프레임 N
Alumni Donation
Name*
E
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

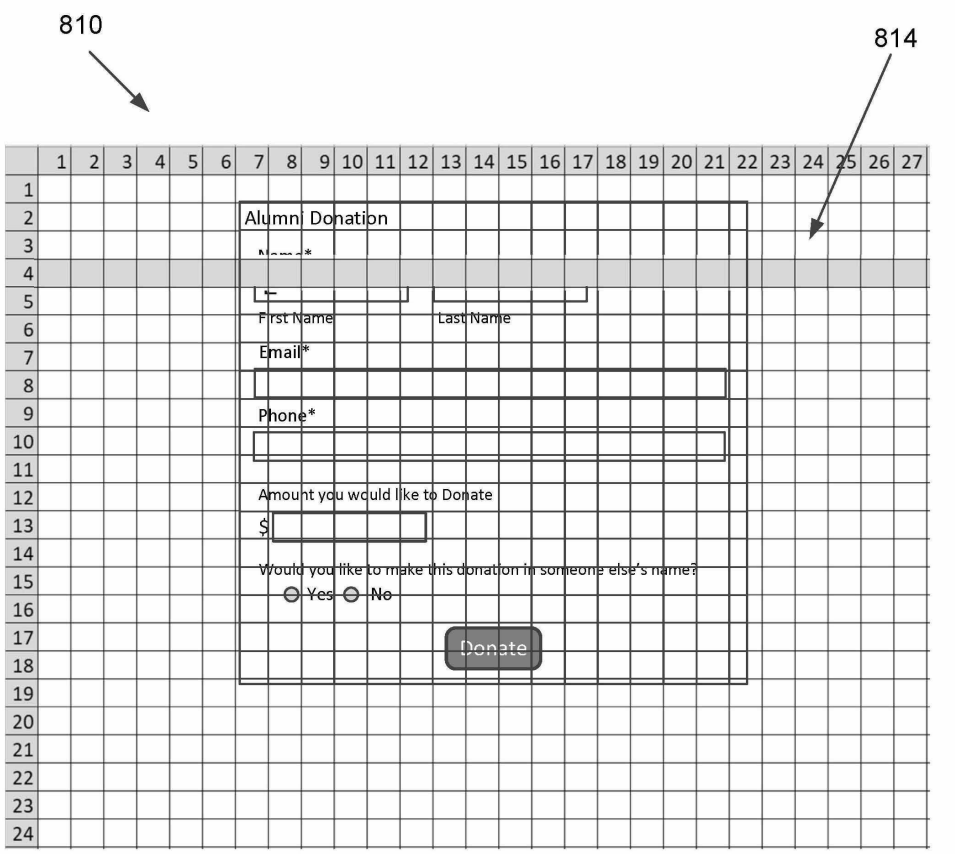
도면8b


800
Alumni Donation
Name*
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

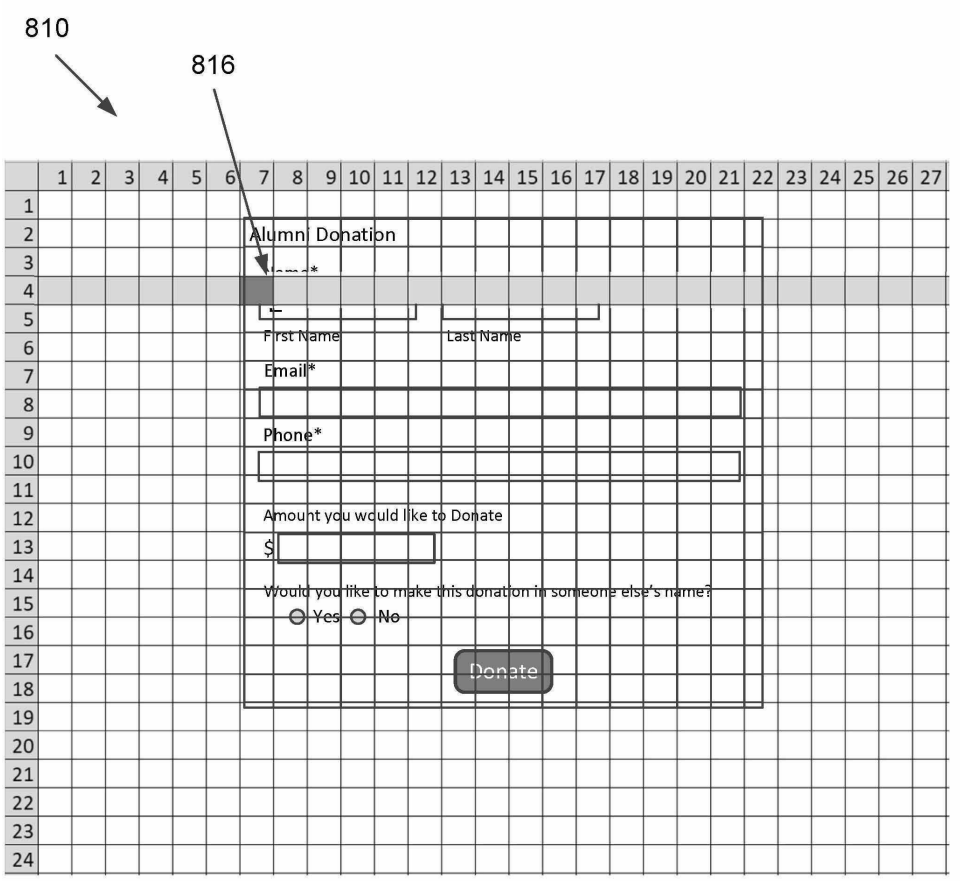
도면8c


810
Alumni Donation
Name*
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

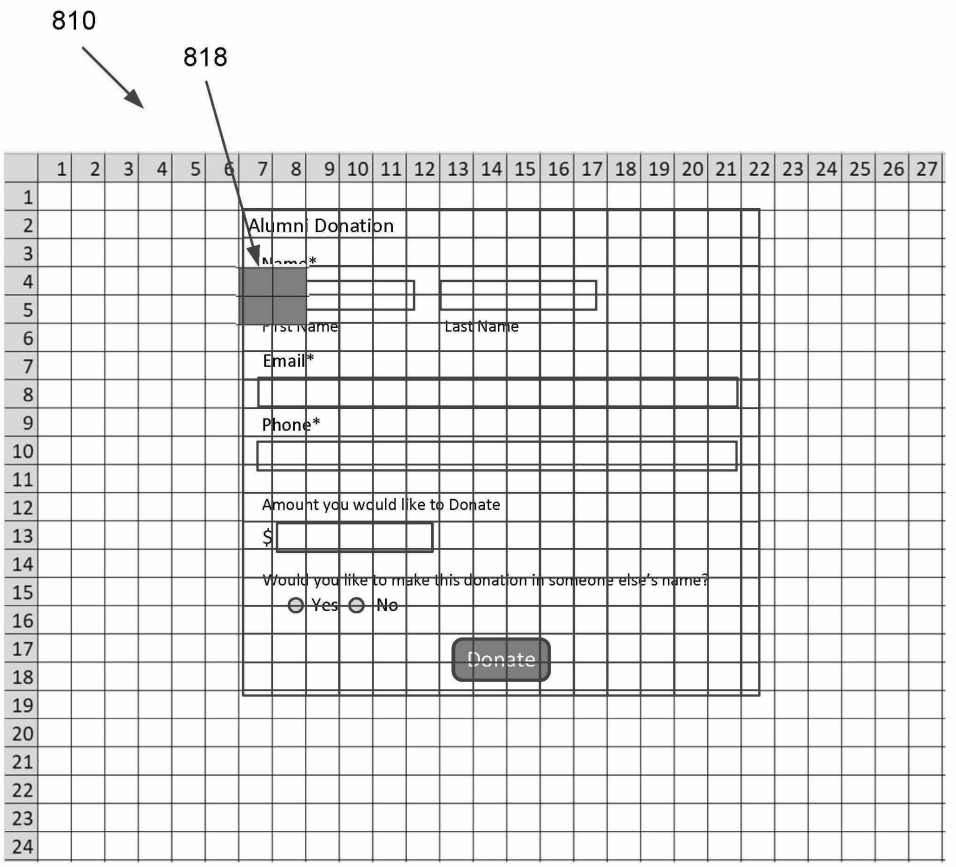
도면8d


810
814
Alumni Donation
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

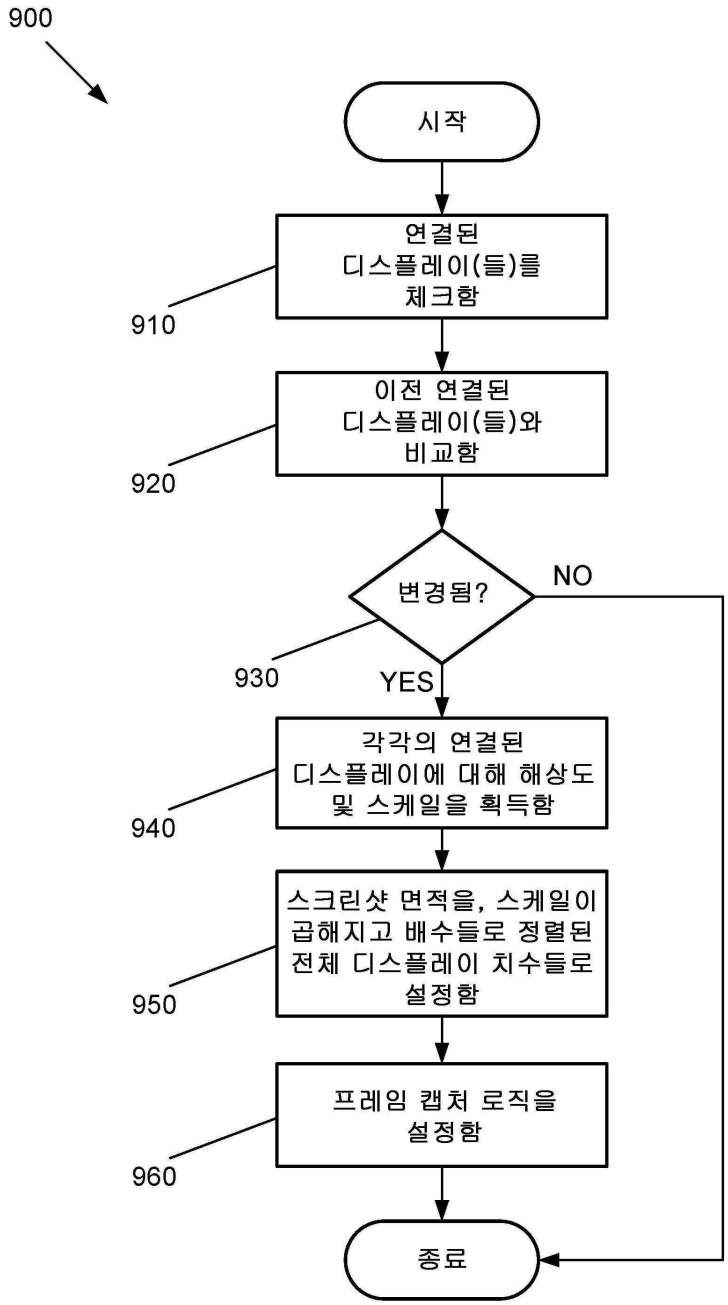
도면8e


810
816
Alumni Donation
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

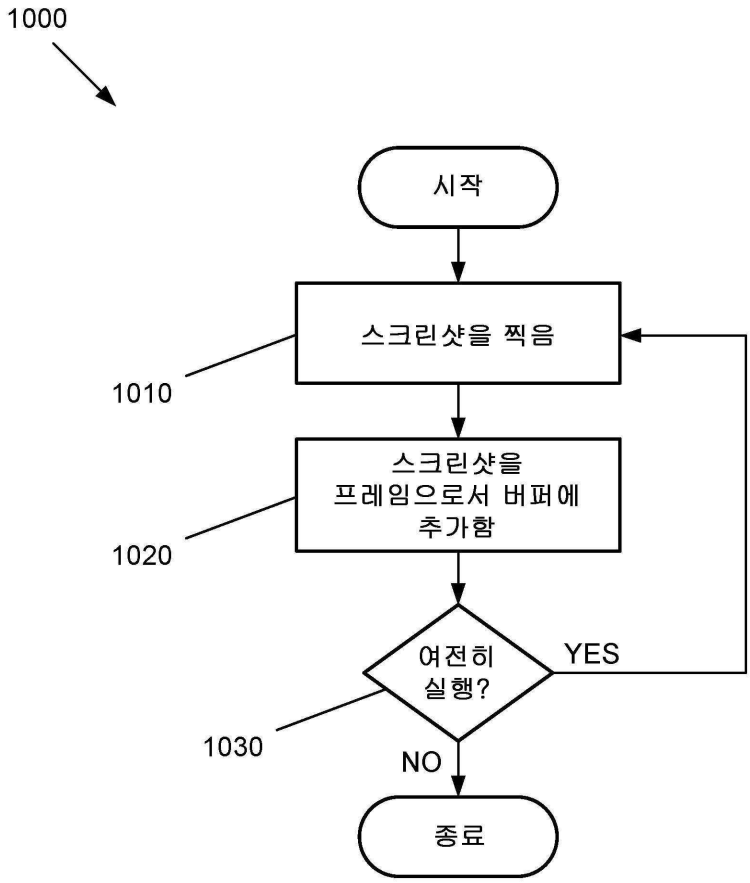
도면8f


810
818
Alumni Donation
First Name
Last Name
Email*
Phone*
Amount you would like to Donate
$
Would you like to make this donation in someone else's name?
Yes
No
Donate

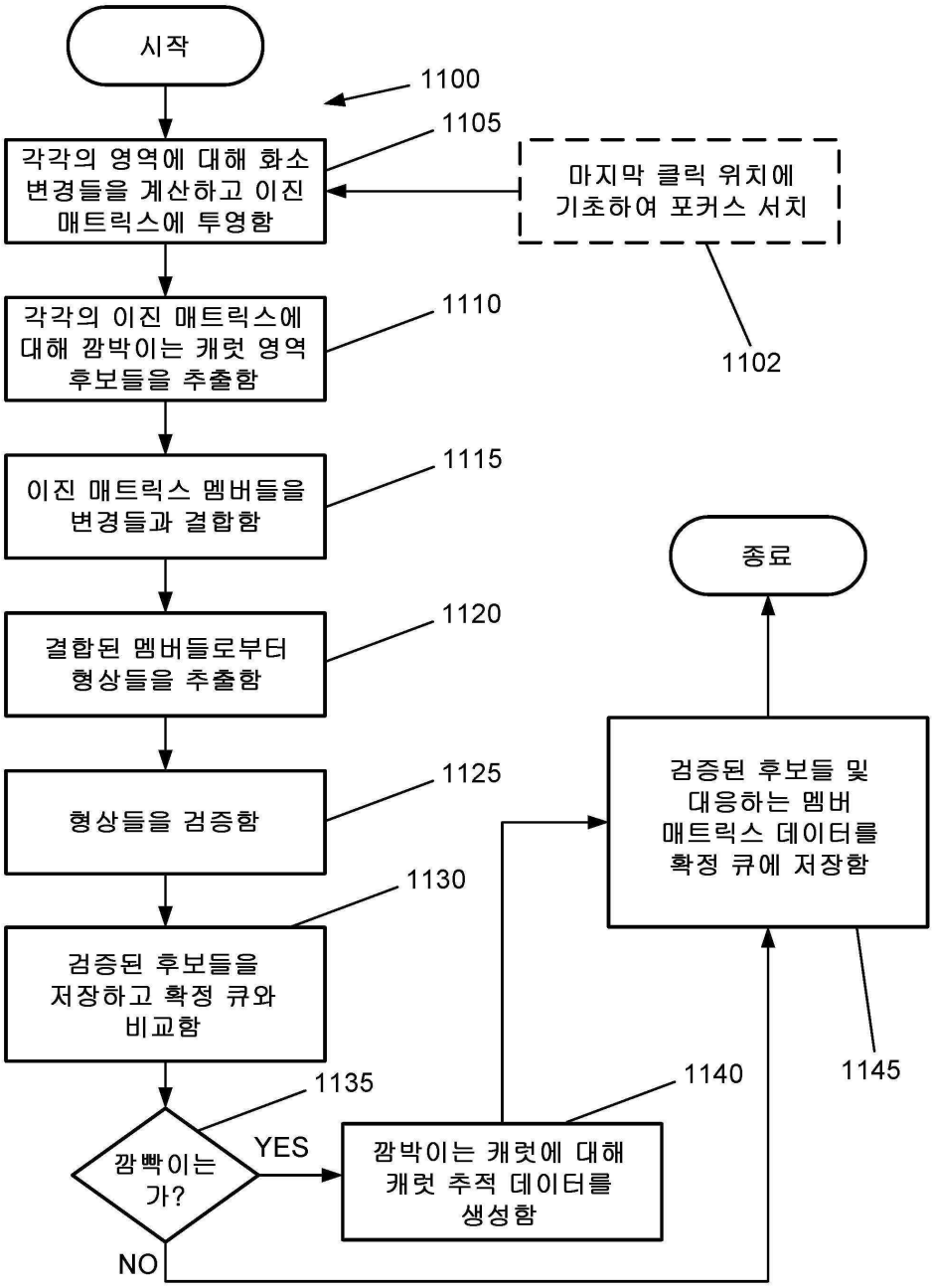
도면9


900
시작
연결된 디스플레이(들)를 체크함
910
이전 연결된 디스플레이(들)와 비교함
920
변경됨?
930
NO
YES
각각의 연결된 디스플레이에 대해 해상도 및 스케일을 획득함
940
스크린샷 면적을, 스케일이 곱해지고 배수들로 정렬된 전체 디스플레이 치수들로 설정함
950
프레임 캡처 로직을 설정함
960
종료

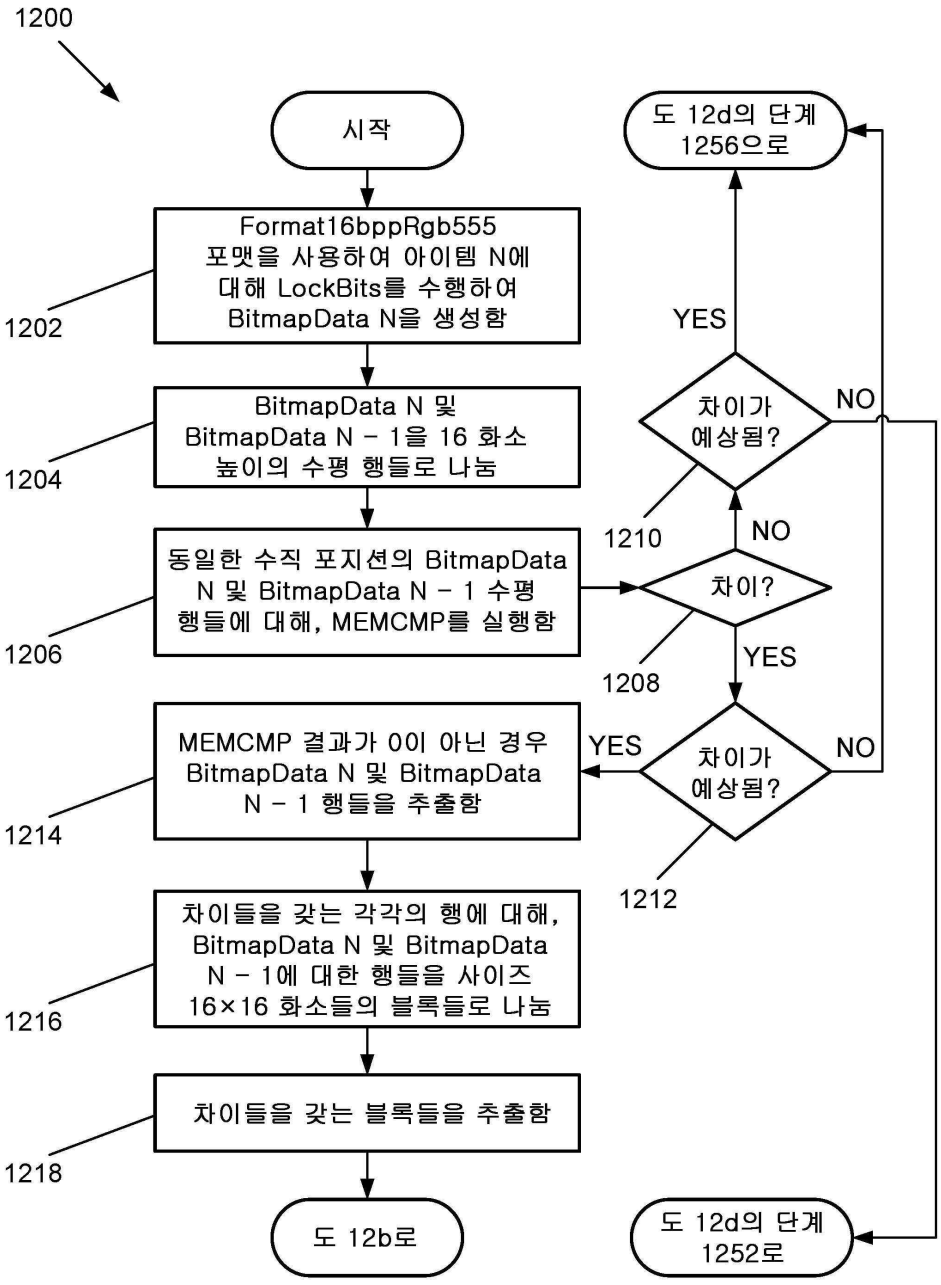
도면10


1000
시작
스크린샷을 찍음
1010
스크린샷을
프레임으로서 버퍼에
추가함
1020
여전히
실행?
YES
1030
NO
종료

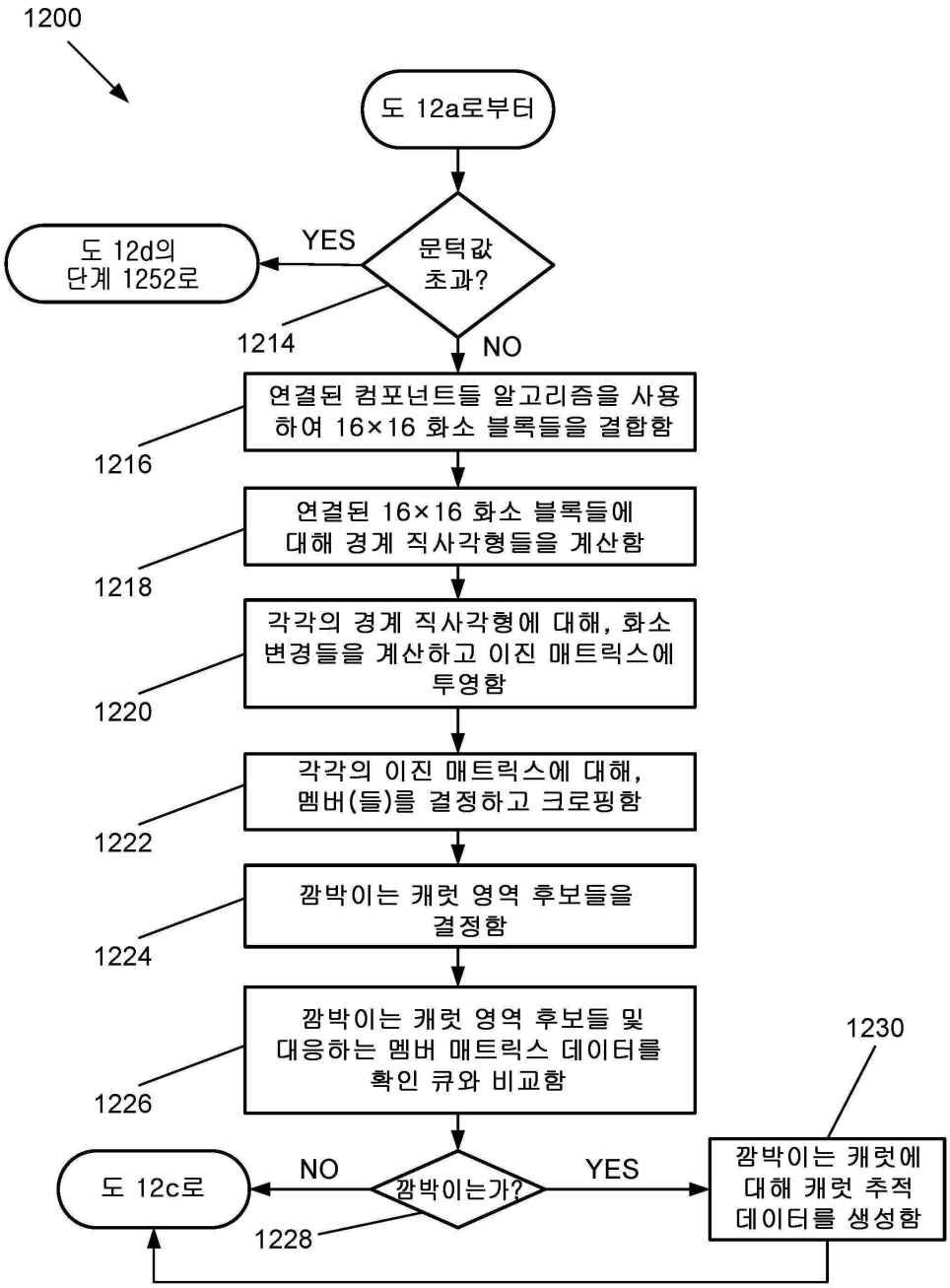
도면11


시작
1100
1105
각각의 영역에 대해 화소 변경들을 계산하고 이진 매트릭스에 투영함
마지막 클릭 위치에 기초하여 포커스 서치
1102
각각의 이진 매트릭스에 대해 깜박이는 캐럿 영역 후보들을 추출함
1110
이진 매트릭스 멤버들을 변경들과 결합함
1115
결합된 멤버들로부터 형상들을 추출함
1120
형상들을 검증함
1125
검증된 후보들을 저장하고 확정 큐와 비교함
1130
깜빡이는가?
1135
YES
NO
깜박이는 캐럿에 대해 캐럿 추적 데이터를 생성함
1140
검증된 후보들 및 대응하는 멤버 매트릭스 데이터를 확정 큐에 저장함
1145
종료

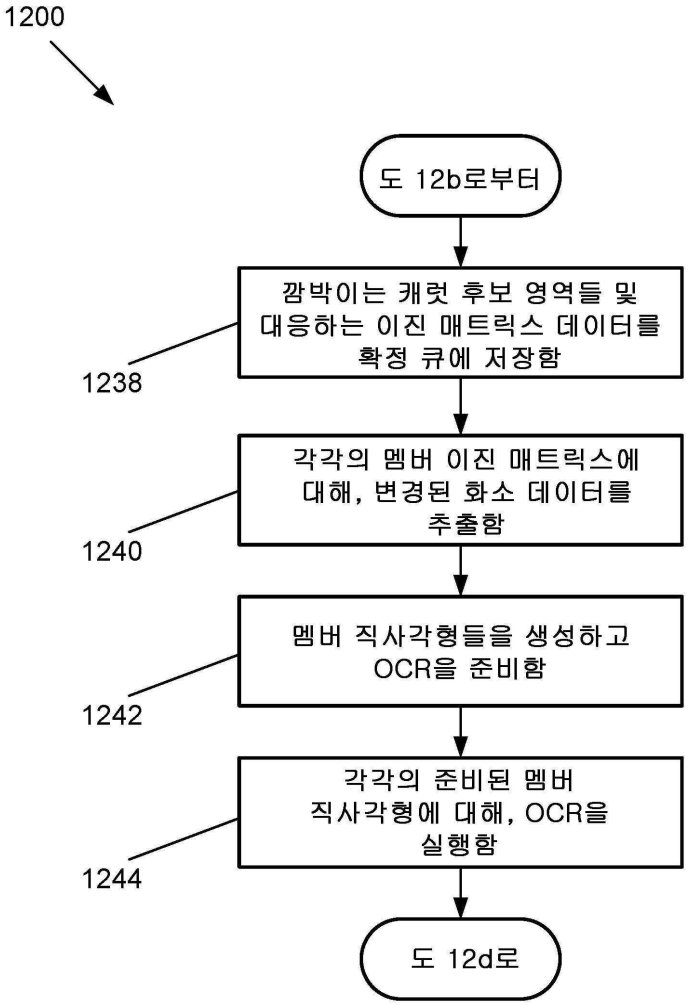
도면12a


1200
시작
Format16bppRgb555 포맷을 사용하여 아이템 N에 대해 LockBits를 수행하여 BitmapData N을 생성함
1202
BitmapData N 및 BitmapData N - 1을 16 화소 높이의 수평 행들로 나눔
1204
동일한 수직 포지션의 BitmapData N 및 BitmapData N - 1 수평 행들에 대해, MEMCMP를 실행함
1206
MEMCMP 결과가 0이 아닌 경우 BitmapData N 및 BitmapData N - 1 행들을 추출함
1214
차이들을 갖는 각각의 행에 대해, BitmapData N 및 BitmapData N - 1에 대한 행들을 사이즈 16×16 화소들의 블록들로 나눔
1216
차이들을 갖는 블록들을 추출함
1218
도 12b로
차이?
1208
YES
NO
차이가 예상됨?
1210
YES
NO
차이가 예상됨?
1212
YES
NO
도 12d의 단계 1256으로
도 12d의 단계 1252로

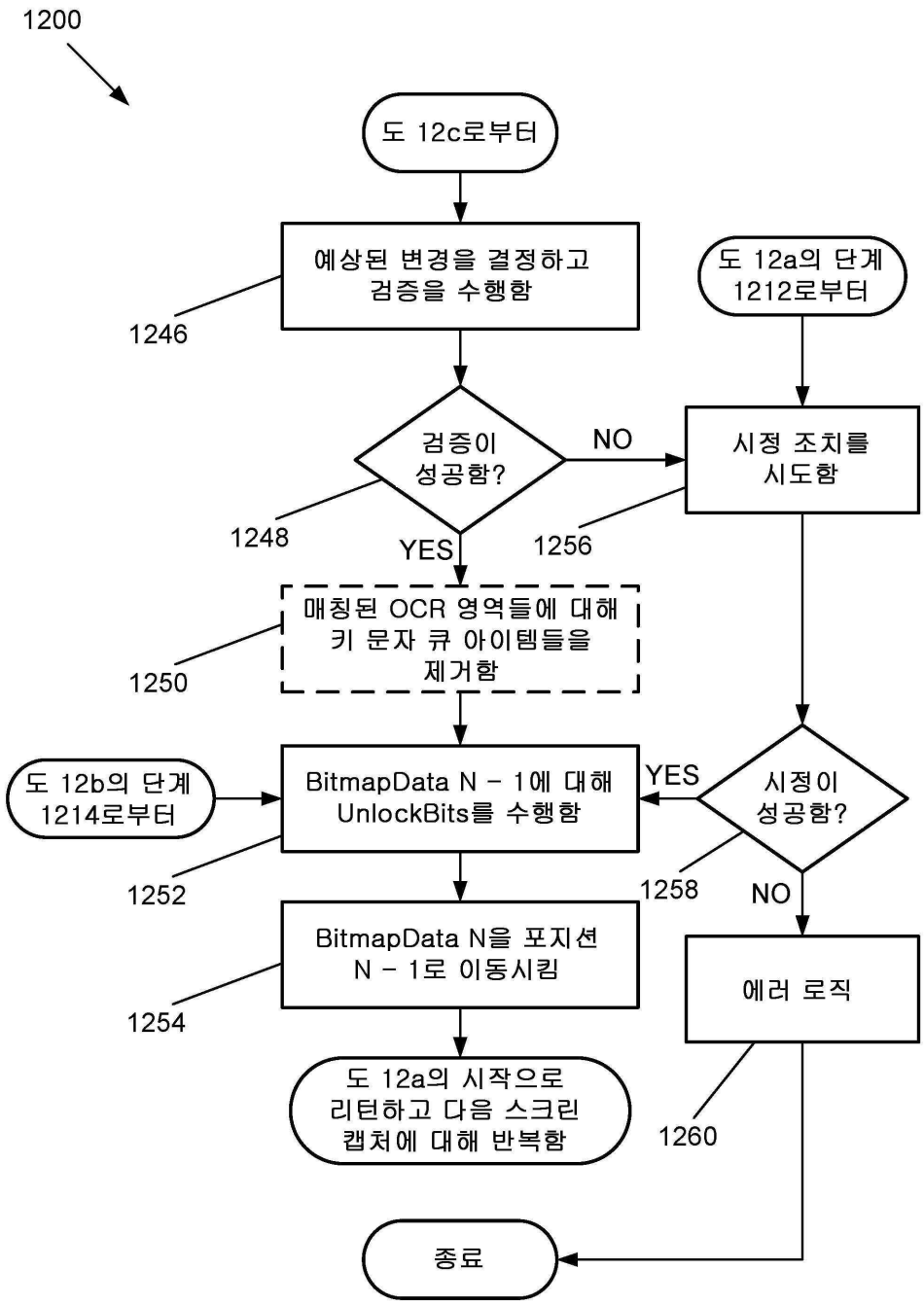
도면12b


1200
도 12a로부터
문턱값 초과?
YES
도 12d의 단계 1252로
1214
NO
연결된 컴포넌트들 알고리즘을 사용하여 16×16 화소 블록들을 결합함
1216
연결된 16×16 화소 블록들에 대해 경계 직사각형들을 계산함
1218
각각의 경계 직사각형에 대해, 화소 변경들을 계산하고 이진 매트릭스에 투영함
1220
각각의 이진 매트릭스에 대해, 멤버(들)를 결정하고 크로핑함
1222
깜박이는 캐럿 영역 후보들을 결정함
1224
깜박이는 캐럿 영역 후보들 및 대응하는 멤버 매트릭스 데이터를 확인 큐와 비교함
1226
1230
깜박이는가?
NO
YES
도 12c로
1228
깜박이는 캐럿에 대해 캐럿 추적 데이터를 생성함

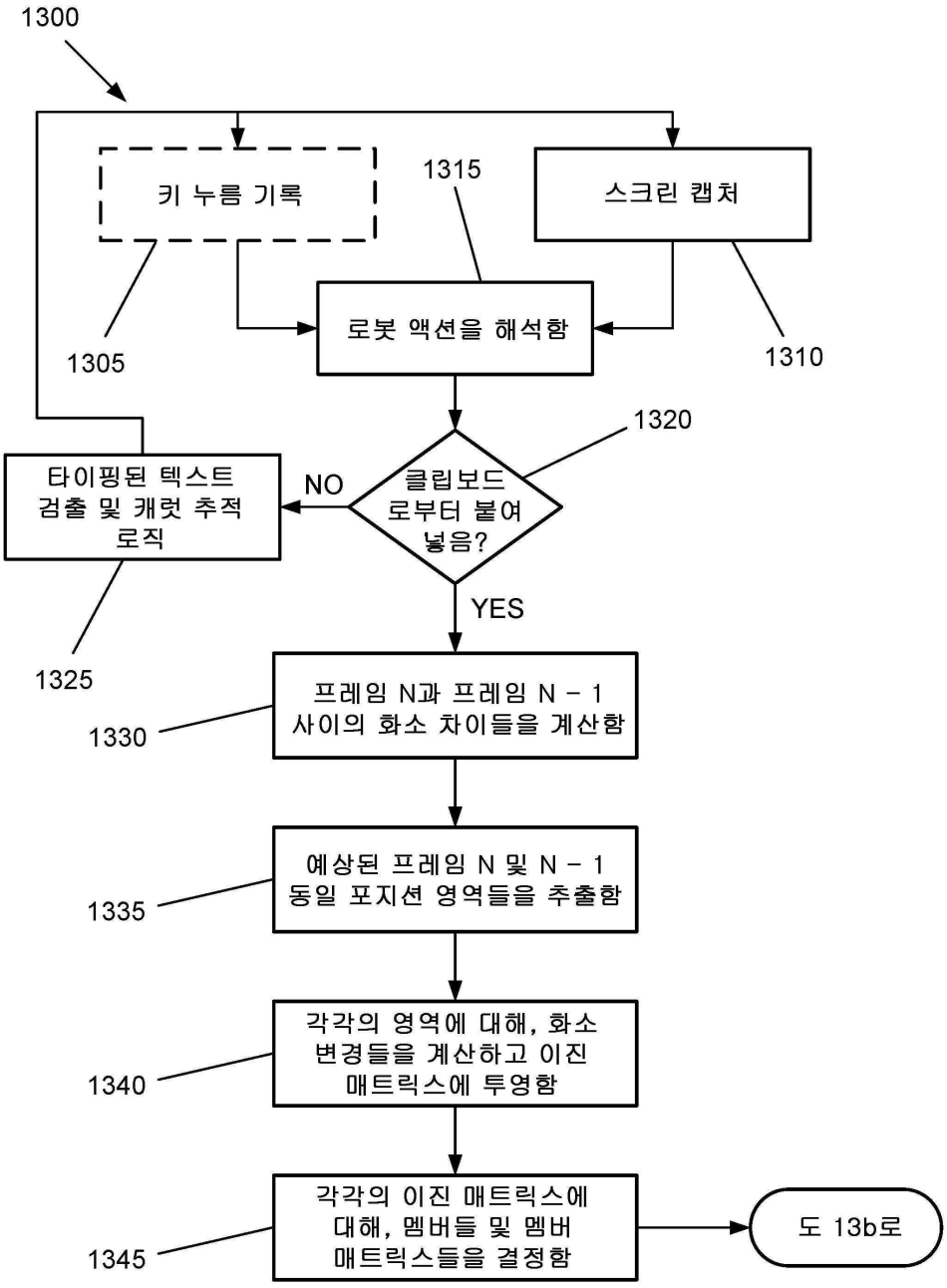
도면12c


1200
도 12b로부터
깜박이는 캐럿 후보 영역들 및 대응하는 이진 매트릭스 데이터를 확정 큐에 저장함
1238
각각의 멤버 이진 매트릭스에 대해, 변경된 화소 데이터를 추출함
1240
멤버 직사각형들을 생성하고 OCR을 준비함
1242
각각의 준비된 멤버 직사각형에 대해, OCR을 실행함
1244
도 12d로

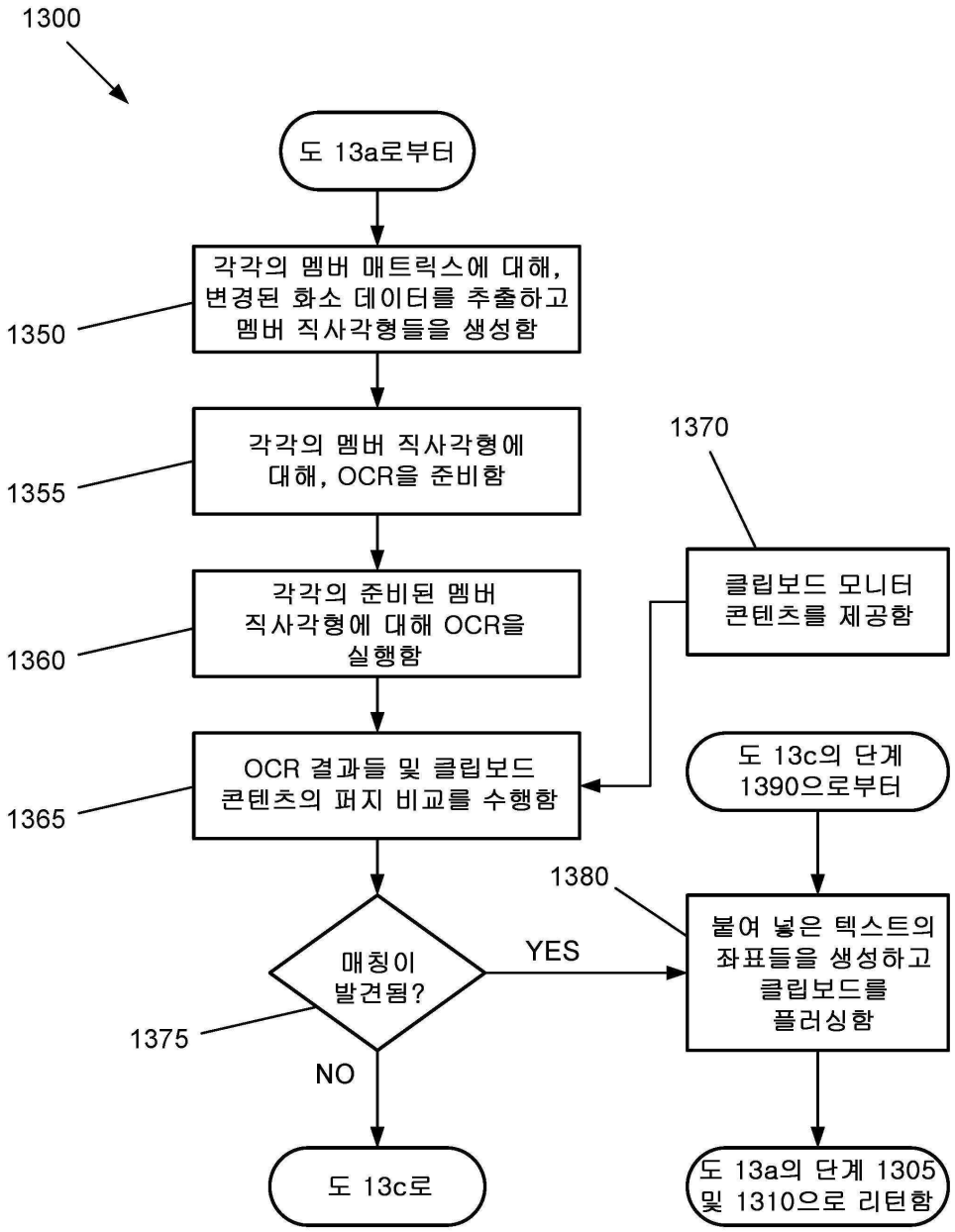
도면12d


1200
도 12c로부터
예상된 변경을 결정하고 검증을 수행함
1246
도 12a의 단계 1212로부터
검증이 성공함?
NO
1248
YES
시정 조치를 시도함
1256
매칭된 OCR 영역들에 대해 키 문자 큐 아이템들을 제거함
1250
도 12b의 단계 1214로부터
BitmapData N - 1에 대해 UnlockBits를 수행함
1252
YES
시정이 성공함?
1258
NO
BitmapData N을 포지션 N - 1로 이동시킴
1254
에러 로직
도 12a의 시작으로 리턴하고 다음 스크린 캡처에 대해 반복함
1260
종료

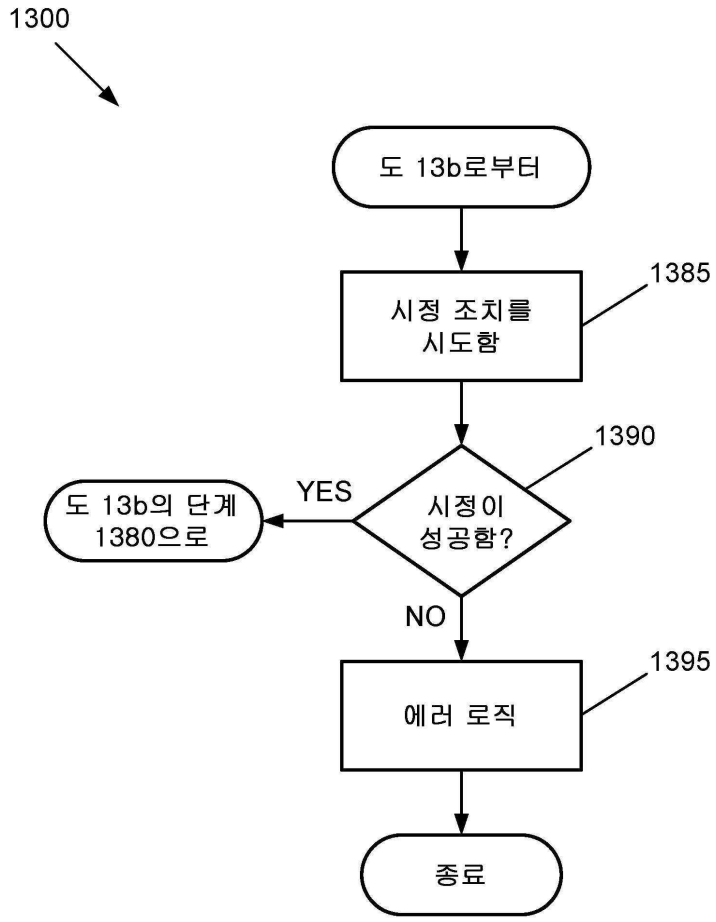
도면13a


1300
키 누름 기록
1315
스크린 캡처
1305
로봇 액션을 해석함
1310
1320
타이핑된 텍스트 검출 및 캐럿 추적 로직
NO
클립보드로부터 붙여 넣음?
YES
1325
1330
프레임 N과 프레임 N - 1 사이의 화소 차이들을 계산함
1335
예상된 프레임 N 및 N - 1 동일 포지션 영역들을 추출함
1340
각각의 영역에 대해, 화소 변경들을 계산하고 이진 매트릭스에 투영함
1345
각각의 이진 매트릭스에 대해, 멤버들 및 멤버 매트릭스들을 결정함
도 13b로

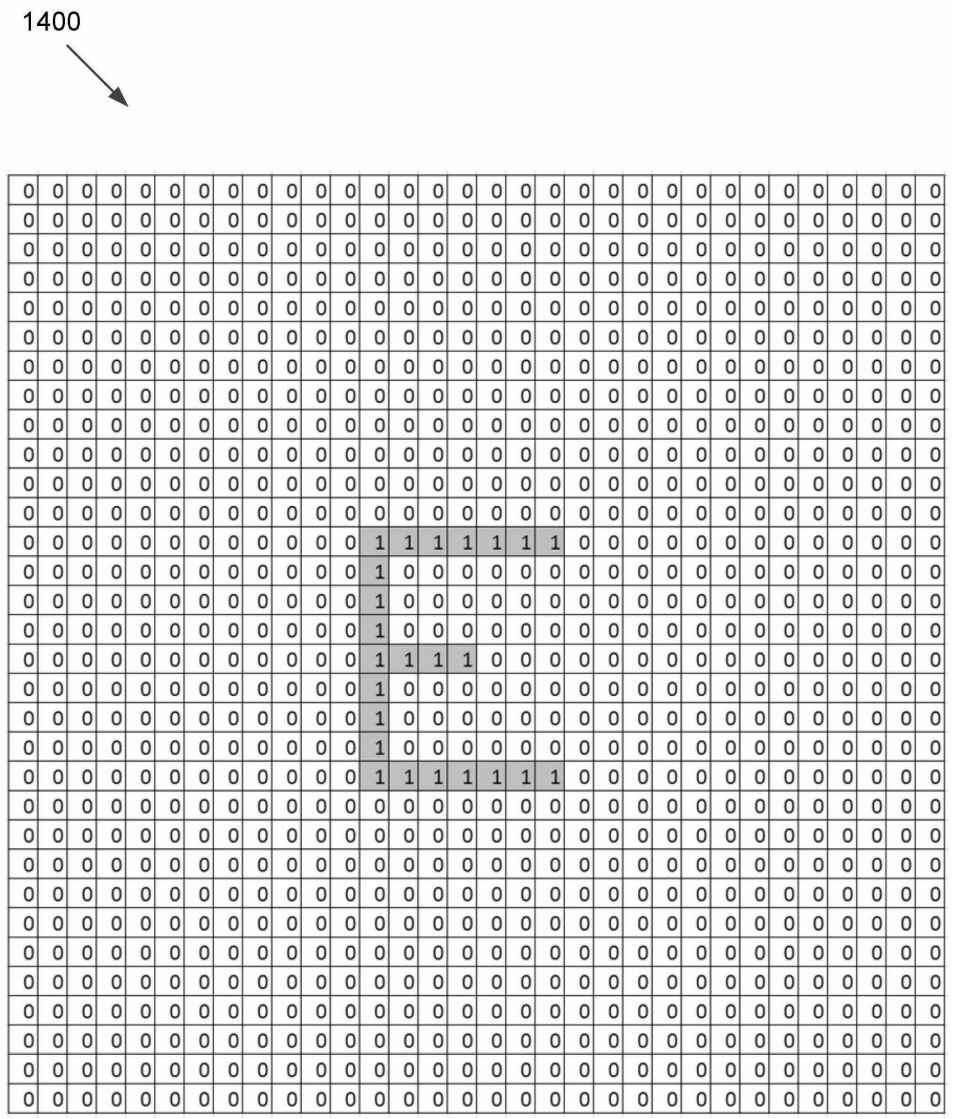
도면13b


1300
도 13a로부터
각각의 멤버 매트릭스에 대해, 변경된 화소 데이터를 추출하고 멤버 직사각형들을 생성함
1350
각각의 멤버 직사각형에 대해, OCR을 준비함
1355
각각의 준비된 멤버 직사각형에 대해 OCR을 실행함
1360
OCR 결과들 및 클립보드 콘텐츠의 퍼지 비교를 수행함
1365
1370
클립보드 모니터 콘텐츠를 제공함
도 13c의 단계 1390으로부터
1380
붙여 넣은 텍스트의 좌표들을 생성하고 클립보드를 플러싱함
매칭이 발견됨?
1375
YES
NO
도 13c로
도 13a의 단계 1305 및 1310으로 리턴함

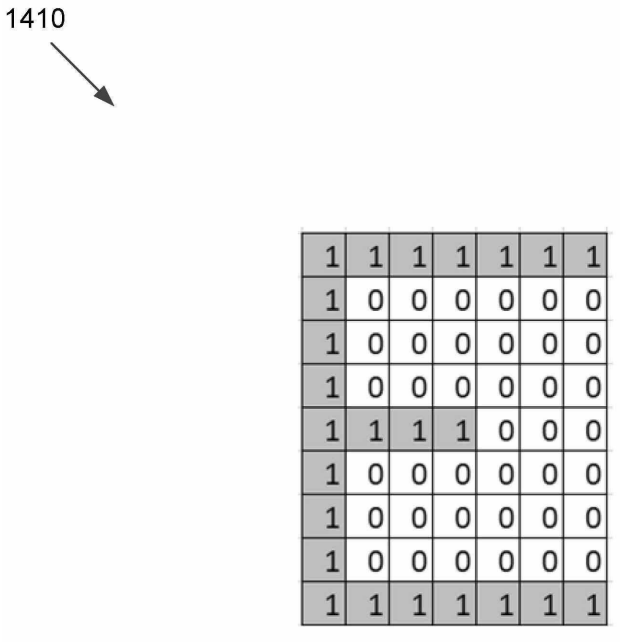
도면13c


1300
도 13b로부터
시정 조치를 시도함
1385
시정이 성공함?
1390
YES
도 13b의 단계 1380으로
NO
에러 로직
1395
종료

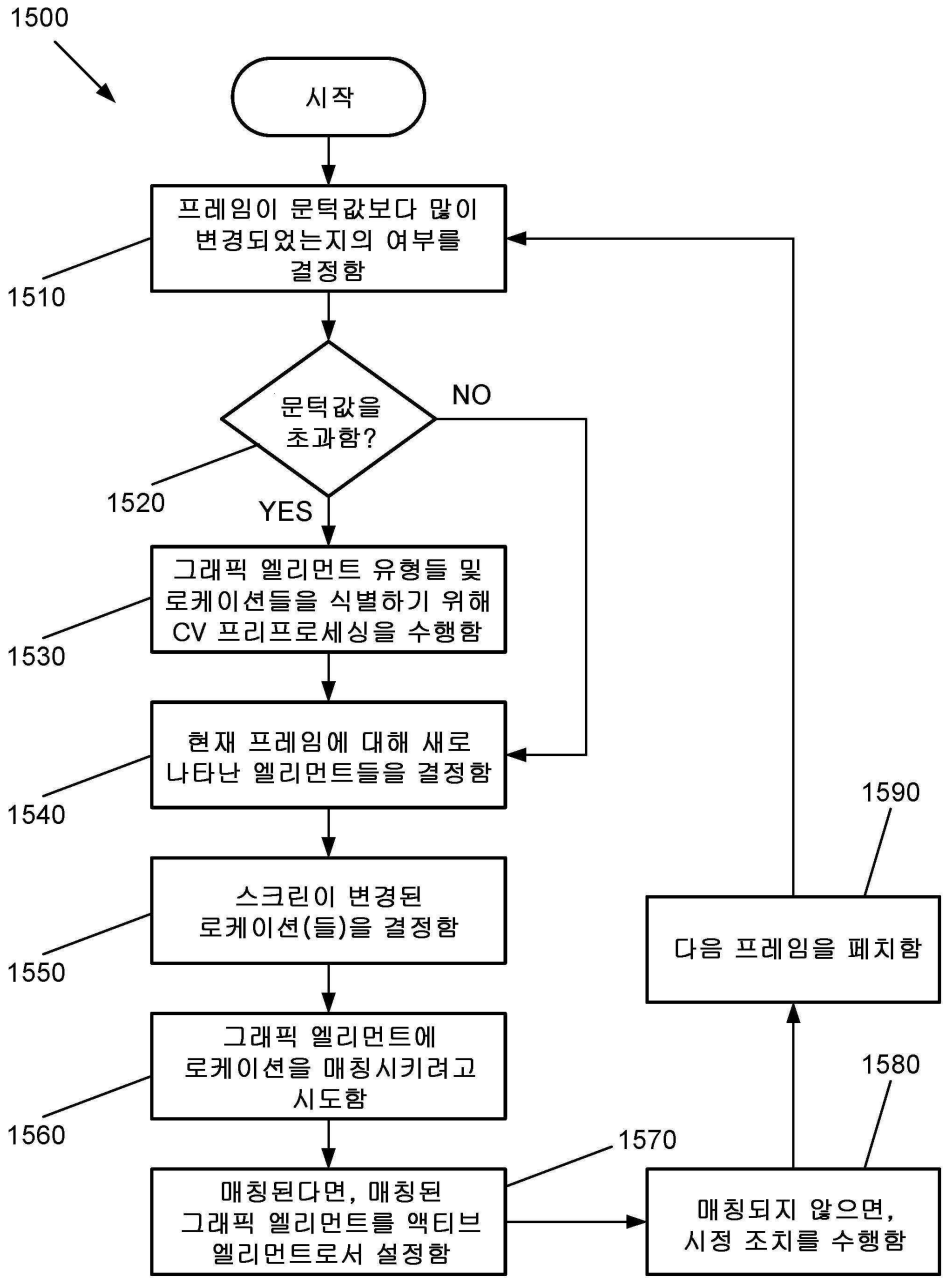
도면14a



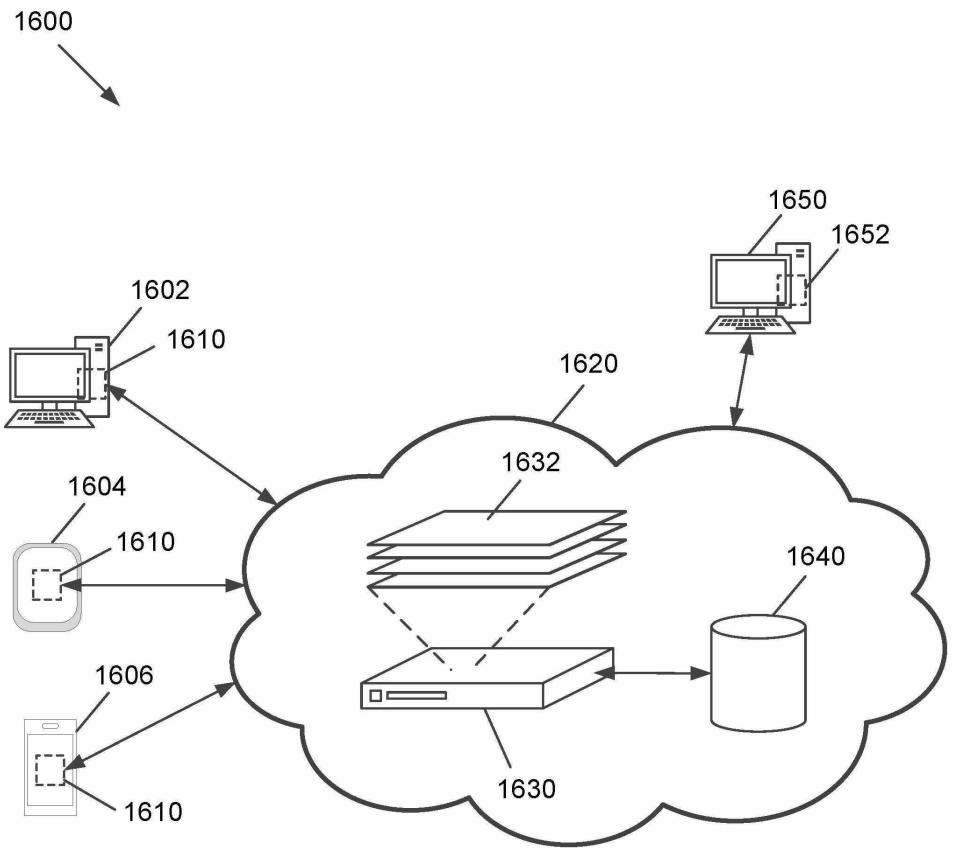
도면14b


1410
1 1 1 1 1 1 1
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 1 1 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 1 1 1 1 1

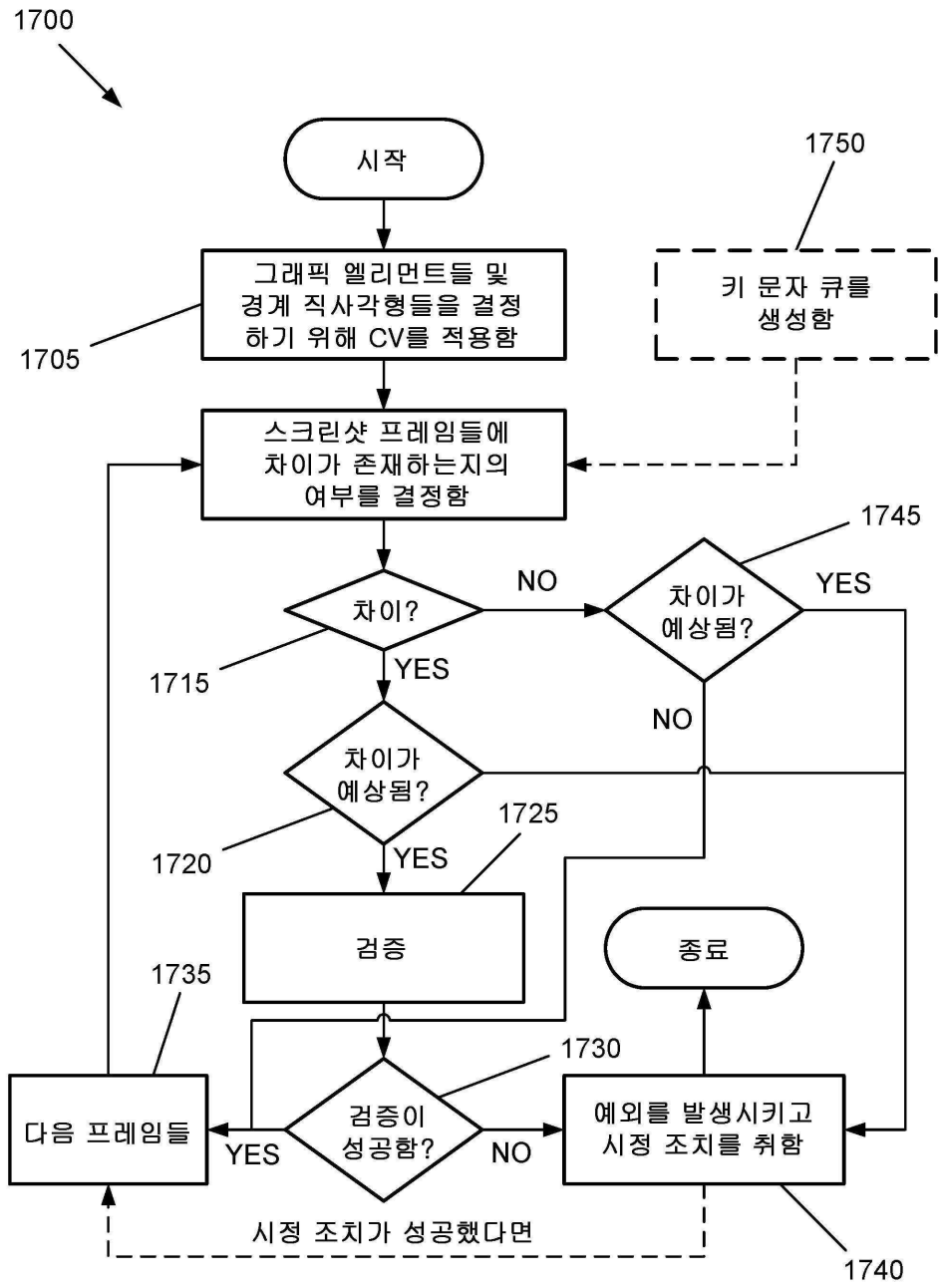
도면15


1500
시작
프레임이 문턱값보다 많이 변경되었는지의 여부를 결정함
1510
문턱값을 초과함?
NO
1520
YES
그래픽 엘리먼트 유형들 및 로케이션들을 식별하기 위해 CV 프리프로세싱을 수행함
1530
현재 프레임에 대해 새로 나타난 엘리먼트들을 결정함
1540
스크린이 변경된 로케이션(들)을 결정함
1550
그래픽 엘리먼트에 로케이션을 매칭시키려고 시도함
1560
매칭된다면, 매칭된 그래픽 엘리먼트를 액티브 엘리먼트로서 설정함
1570
매칭되지 않으면, 시정 조치를 수행함
1580
다음 프레임을 페치함
1590

도면16


1600
1650
1652
1602
1610
1620
1632
1604
1610
1640
1606
1610
1630

도면17


1700
시작
1750
그래픽 엘리먼트들 및 경계 직사각형들을 결정하기 위해 CV를 적용함
1705
키 문자 큐를 생성함
스크린샷 프레임들에 차이가 존재하는지의 여부를 결정함
1745
차이?
NO
차이가 예상됨?
YES
1715
YES
NO
차이가 예상됨?
1725
1720
YES
검증
종료
1735
1730
다음 프레임들
YES
검증이 성공함?
NO
예외를 발생시키고 시정 조치를 취함
시정 조치가 성공했다면
1740