PŘIHLÁŠKA VYNÁLEZU

zveřejněná podle § 31 zákona č. 527/1990 Sb.

(21) Číslo dokumentu:

2077-97

(19) ČESKÁ REPUBLIKA

ÚŘAD PRŮMYSLOVÉHO VLASTNICTVÍ

(22) Přihlášeno: **28. 12. 95**

(32) Datum podání prioritní přihlášky: **04.01.95**

(31) Číslo prioritní přihlášky: **95/368644**

(33) Země priority: **US**

(40) Datum zveřejnění přihlášky vynálezu: **14. 04. 99** **(Věstník č. 4/99)**

(86) PCT číslo: **PCT/CA95/00727**

(87) PCT číslo zveřejnění: **WO 96/21171**

(13) Druh dokumentu: **A3**

(51) Int. Cl.[6]:

G 02 B 27/22
G 02 B 3/00
H 04 N 13/00
H 04 N 13/04

(71) Přihlášovatel:
VISUALABS INC., Calgary, CA;

(72) Původce:
Zelitt Sheldon S., Calgary, CA;

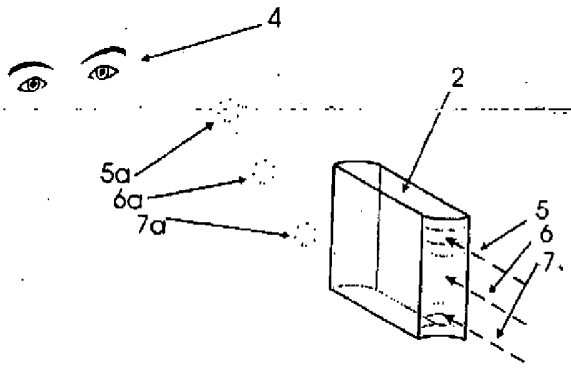
(74) Zástupce:
Kania František Ing., Mendlovo nám. 1a, Brno, 60300;

(54) Název přihlášky vynálezu:

Způsob vytváření trojrozměrného obrazu ze zobrazení dvojrozměrného obrazu a zařízení k provádění tohoto způsobu

(57) Anotace:

Způsob vytváření trojrozměrného obrazu ze zobrazení dvojrozměrného obrazu, vytvořeného diskrétními obrazovými prvky, obsahující zajištění soustavy optických prvků, příslušně scentrovaných před obrazovými prvky, a měnění efektivní ohniskové vzdálenosti každého optického prvku pro změnu zdánlivé vizuální vzdálenosti od pozorovatele, nacházejícího se před zobrazovačem, na němž je znázorněn každý jednotlivý obrazový bod. Podstata vynálezu spočívá v tom, že každý optický prvek má ohniskovou délku, která se mění progresivně podél ploch orientovaných obecně rovnoběžně s obrazem, přičemž měnění efektivní ohniskové délky každého optického prvku obsahuje kroky posuvu místa, na němž je světlo emitováno z dvourozměrného obrazu bezprostředně uvnitř každého obrazového prvku, a převádění emitovaného světla k optickým prvkům, kde místo, na nějž emitované světlo na optických prvcích dopadá, určuje zdánlivou hloubku obrazového bodu. Zobrazovací zařízení pro provádění tohoto způsobu je opatřeno prostředky /18, 65/ pro malý posuv místa /5b, 6b, 7b/, v němž je světlo v obrazovém bodě emitováno, a to podle požadované hloubky tak, že je zde odpovídající posuv místa /5, 6, 7/ vstupu světla podél vstupní plochy optického prvku, čímž se účinná ohnisková délka dynamicky mění a zdánlivá vizuální vzdálenost /5a, 6a, 7a/ od pozorovatele se mění podle posuvu místa vstupu světla.



CZ 2077-97 A3

Zobrazení dvojrozměrného obrazu a zařízení k provádění tohoto způsobu

Oblast techniky

Vynález se týká techniky zobrazování trojrozměrného obrazu, a to zejména takové techniky, při níž se nevyžaduje používání speciálních helem nebo brýlí.

Dosavadní stav techniky

Vytvoření plného trojrozměrného obrazu bylo vážným technologickým cílem po značnou část dvacátého století. Již v roce 1908 Gabriel Lippman vynalezl způsob vytváření pravdivého trojrozměrného obrazu scény používajícího fotografickou desku exponovanou přes "muší oko" vytvořené čočkovou fólií z malých pevných čoček. Tato technika se stala známou jako integrální fotografie a zobrazování takto vytvořených obrazů se provádělo přes čočkovou fólii z fixních čoček téhož druhu. Lippmanův vývoj a jeho pokračování v následujících letech, viz např. US patent č. 3,878,329, však nedokázalo vytvořit technologii, která by byla přímo použitelná pro vytváření obrazů, které by šlo jednoduše vyrábět, které by byly adaptovatelné na pohyblivou prezentaci nebo by byly schopné přímo reprodukovat elektronicky generované obrazy, což je převládající formát této poslední části tohoto století.

V průběhu času docházelo k dalšímu rozpracovávání trojrozměrného zobrazování formou vytváření obrazu ze soustavy obrazových složek, což vyústilo do různých technických zlepšení, která zahrnují různá provedení žebrových, čočkových nebo mřížkových folií optických prvků pro výrobu stereoobra-

zů z jediného speciálně zpracovaného obrazu, viz např. US patent č. 4,957,311 nebo US patent č. 4,759,017, aby byly citovány pouze poslední relevantní příklady. Většina z těchto řešení trpí společnými soustavami nedostatků, které zahrnují přísná omezení fyzické polohy pozorovatele vůči pozorovacímu stínítku, sníženou kvalitu obrazu způsobenou rozdělením intenzity vytvářeného obrazu mezi dva oddělené obrazy a v mnoha řešeních paralaxa pozorovatelná v pouze jednom směru.

Další dosud známé techniky pro generování skutečných trojrozměrných obrazů zahrnují rastrování fyzikálního objemu, a to buď mechanickým rastrováním laserového paprsku přes rotující spirálové stínítko nebo difuzní oblak par, sekvenčním aktivováním soustavy vnitřních fosforových stínítek v obrazovce nebo fyzikálním odchylováním zakřiveného zrcadla pro vytvoření verze konvenčního zařízení pro vytváření obrazu s proměnným ohniskem. Všechny tyto techniky jsou však neohrabané, obtížné jak z hlediska výroby, tak z hlediska pozorování, a celkově je není možno upravit tak, aby je bylo možno umístit na běžný trh.

V průběhu téže časové periody se objevila soustava technologií týkajících se zařízení, která na sobě nesl pozorovatel, a to včetně brýlí používajících dvoubarevných nebo křížově polarizovaných filtrů pro oddělení současně zobrazovaných duálních obrazů, a helmy zobrazující virtuální realitu, z nichž všechny se vztahují na výrobu stereopsie, to jest příjmu vjemu hloubky prostřednictvím spojení odděleného obrazu levého a pravého oka. Některé z těchto technik vytvářely stereoobrazy pozoruhodné kvality, ačkoliv obecně na

účet pohodlí pozorovatele, napětí očí, jasu obrazu a akceptovatelnosti částí pozorující populace, která nemůže přímo a pohodlně přijímat takové stereozobrazení. Toto je prokázáno v nedávno se objevivších oftalmologických a neurologických studiích, které hovoří o škodlivých a potenciálně dlouho trvajících účincích dlouhého používání stereozobrazovacích systémů, ať už je na sobě má pozorovatel nebo jsou jiného typu.

Japonský patentový spis číslo 620 77794 popisuje dvojrozměrný zobrazovač, na němž je obraz vytvořen diskrétními obrazovými body, přičemž zobrazovač je opatřen soustavou optických prvků scentrovaných příslušně před obrazovými body, a prostředky pro individuální změnu účinné ohniskové délky každého optického prvku pro změnu zdánlivé vizuální vzdálenosti od pozorovatele, nacházejícího se před zobrazovačem, na němž je každý jednotlivý obrazový bod znázorněn, čímž se vytváří trojrozměrný obraz.

Optické prvky v tomto japonském patentu jsou čočky vytvořené z nemagnetických tekutých krystalů, přičemž ohnisková délka čoček může být měněna změnou elektrického pole, které mění scentrování krystalů. Tento systém vyžaduje tranzistory a další elektrické spoje nasměrované ke každé mikročočce, a speciální utěsnění mezi skleněnými deskami je nezbytné. Navíc, dosažená změna efektivní ohniskové délky je velmi malá a vyžaduje použití přídavných optických prvků, jako jsou velká zvětšovací skla, což činí systém nepřípustně veliký a nevhodně omezuje příčný pozorovací úhel obrazu, který je k dispozici.

Podstata vynálezu

Předmětem tohoto vynálezu je zajistit zlepšený zobrazovač trojrozměrného obrazu, v němž by byly překonány nedostatky systému popsaného ve výše zmíněné japonské publikaci.

Toho je dosaženo tím, že každý optický prvek má ohniskovou délku, která se progresivně mění podél ploch orientovaných obecně rovnoběžně s obrazem, kde podstata vynálezu spočívá v tom, že zařízení je opatřeno prostředkem pro malý posun místa, v němž je světlo uvnitř obrazového bodu emitováno, a to podle požadované hloubky tak, že je zde odpovídající posuv místa vstupu světla podél vstupní plochy optického prvku, čímž je efektivní ohnisková délka dynamicky měněna a zdánlivá vizuální vzdálenost od pozorovatele se mění podle posuvu místa vstupu světla.

V jednom výhodném provedení jsou optické prvky vytvořeny jako jedna nebo více čoček, ale mohou být namísto toho vytvořeny ze zrcadel nebo případně z kombinace lámavých a odrazových ploch.

Ve své nejjednodušší formě jsou obrazové body a překrývající optické prvky pravoúhlé a ohnisková vzdálenost každého optického prvku se mění progresivně podél délky optického prvku. V tomto případě bod vstupu světla je posunut lineárně podél délky. Avšak i jiné tvary optických prvků a typů posuvu jsou v rozsahu vynálezu. Například optické prvky mohou být kruhové a mít ohniskovou délku, která se mění radiálně vzhledem ke středové optické ose. V takovém případě světlo vstupuje jako plošky mezikruží, které jsou posouvány radiálně.

Stejně tak, ačkoliv změny optických charakteristik

v optických prvcích na úrovni obrazového bodu jsou zde prezentovány jako by byly způsobeny změnami ve tvaru povrchu fyzikálních prvků, bylo v laboratoři úspěšně experimentováno s vytvářením takových variací optických charakteristik prostřednictvím použití optických materiálů s proměnným indexem, kde index lomu se podél optického prvku progresivně mění.

Vztah mezi ohniskovou délkou a posuvem může být lineární nebo nelineární.

Pro zajištění vstupu světla na úrovni obrazového bodu do soustavy optik na úrovni obrazového bodu lze použít soustavy přístrojů. V jednom příkladném provedení tohoto vynálezu je tímto přístrojem pro vstup světla obrazovka umístěná za soustavou optik tak, že světelná čára může být rozmítána horizontálně za každou řadou optik na úrovni obrazového bodu a představována na nepatrně odlišném vertikálním posunu od rozmítané čáry při průchodu za každou optikou. V různých příkladných provedeních může být přístroj pro vstup světla plochý panelový zobrazovací přístroj používající technologii jako jsou tekuté krystaly, elektroluminiscence nebo plasmové zobrazovací přístroje. Elektroluminiscenční přístroje zahrnují soustavy světlo emitujících diod. Ve všech těchto příkladných provedeních je zobrazování pohybu představováno rozmítáním celých obrazů následně zhruba stejným způsobem jako u běžných dvourozměrných zobrazování pohybu. V tomto způsobu zobrazování pohybu může být představováno na četnostech rámců omezených pouze schopností rozmítaného světelného paprsku být nepatrně vertikálně manipulován pro každý obrazový bod. Ačkoliv v žádném případě technologie není tímto limitována, příkladné provedení tohoto vynálezu tak jek je zde popsáno bylo úspěšně prováděno v laboratořích přihlašovatele na četnostech rámců v rozsahu až 111 rámců za sekundu.

V ještě dalším výhodném provedení vynálezu může celé osvětlení obrazu na úrovni obrazových bodů přijít ze speciálně připraveného filmu zachycujícího pohyblivý obraz nebo

ze speciálně připraveného nehybného fotografického filmu, v němž každý rámec nebo film je osvětlován zezadu běžným způsobem, ale pozorován soustavou optik téhož typu na úrovni obrazových bodů . V tomto příkladném provedení každý přenášený světelný obrazový bod v každém průsvitném rámci je umístěn specificky podél povrchu lineárního vstupu optik tak, že jeho vertikální vstupní bod generuje světelný bod umístěný ve specifické vzdálenosti od pozorovatele, v níž se požaduje příjem tohoto daného obrazového bodu, stejně tak jako v elektronicky osvětlených příkladných provedeních, které byly popsány výše. Takové běžně známé systémy zahrnují promítání trojrozměrného zobrazení do volného prostoru odrazem od konkávního zrcadla nebo optiky s podobným promítáním obrazu. Tato technika je značně přesvědčivější než promítání konvenčního plochého dvojrozměrného zobrazení v tom, že promítaný trojrozměrný zobrazovaný obraz ve volném prostoru má ve skutečnosti reálnou pozorovatelnou hloubku. Doposud byla úspěšně použita konkávní zrcadla se zakřivením sférickým, parabolickým a hyperbolickým, ale i jiné konkávní tvary jsou zřejmě možné.

Ve všech těchto příkladných provedeních může být trojrozměrný obraz pozorován přímo nebo použit jako zdroj reálného obrazu pro jakýkoliv běžně známý systém promítání reálného obrazu.

Přehled obrázků na výkresech

Tyto a další předměty a význaky tohoto vynálezu budou zjevnější z následujícího popisu ve spojení s připojenými výkresy. V těchto výkresech budou podobné součásti označeny

podobnými vztahovými značkami. Na obrázku 1 (a) je zobrazení jednoho příkladného provedení optického přístroje na úrovni obrazového bodu v pohledu zešikma zezadu, obrázek 1(b) je zobrazením odlišného příkladného provedení téhož typu optické soustavy na úrovni obrazového bodu, obsahujícího tři optické prvky, obrázek 2 znázorňuje způsob, kterým změna bodu vstupu kolimovaného světelného paprsku do zadní strany optického přístroje (vstupní konec) mění vzdálenost v prostoru od pozorovatele, v níž se tento světelný bod objevuje, obrázek 3(a) znázorňuje, jak toto proměnné vstupní osvětlení optického přístroje na úrovni obrazového bodu může být v jednom příkladném provedení zajištěno obrazovkou, obrázek 3 (b) znázorňuje odlišný pohled na proměnné vstupní osvětlení a scentrování optiky na úrovni obrazového bodu s obrazovými body na fosforové vrstvě obrazovky, obrázek 3(c) zobrazuje vztah mezi velikostí a poměrem stran kolimovaného vstupního světelného paprsku k velikosti a poměru stran optického přístroje na úrovni obrazového bodu, obrázek 4 (a) znázorňuje, jak je soustava optik na úrovni obrazového bodu představována před zdrojem osvětlení, jakým je obrazovka monitoru počítače, televizor nebo jiný v podstatě plochý stínítkový zobrazovací přístroj, obrázek 4 (b) znázorňuje druhé výhodné příkladné provedení vzorce obrazových bodů obrazovky, které může být použito pro tento účel, obr. 5 znázorňuje způsob jakým je signál hloubky přidáván do horizontálně rozmítaných rastrových řádků v obrazu televizního nebo počítačového monitoru, obr. 6 znázorňuje jak může být specifický bod vstupu světla do optiky na úrovni obrazových bodů měněn za použití filmu s pohyblivými obrazy nebo nějaké jiné formy osvětlené-

ho průsvitného předmětu jako zdroje osvětlení, obr. 7 znázorňuje, jak lze použít soustavu optik na úrovni obrazového bodu pro sledování průběžného pásku filmu s pohyblivými obrazy pro sledování následných rámců filmu v zobrazení trojrozměrných pohyblivých obrazů, obr. 8 zobrazuje způsob v němž hloubková složka zaznamenaného obrazu může být odvozena prostřednictvím zachycení obrazu používajícího jednu hlavní zobrazovací kameru a jednu sekundární kameru, obrázek 9 (a) znázorňuje proces, kterým signál hloubky může být retroaktivně odvozen pro konvenční dvourozměrné zobrazení, čímž je tento obraz schopen trojrozměrného zobrazení na vhodném zobrazovacím přístroji, obrázek 9 (d) znázorňuje propojení a činnost obraz zpracujících přístrojů, které lze použít pro přidání hloubky do videozobrazení podle postupu znázorněného na obrázku 9(a), obr. 10 znázorňuje aplikaci technik hloubkového zobrazení na úrovni obrazového bodu odvozenou v průběhu tohoto vývoje na trojrozměrné zobrazení tištěných obrazů, obr. 11 znázorňuje rozložení energie běžného obrazového signálu NTSC indikující nosné jasu a barvy, obr. 12 znázorňuje totéž rozložení energie obrazového signálu NTSC, ale se signálem hloubky zakódovaným do spektra, obrázek 13 (a) znázorňuje funkční návrh obvodu v běžném televizním přijímači, který typicky řídí vertikální vychýlení rozmítaného elektronového svazku v obrazovce, obrázek 13 (b) znázorňuje tytéž obvody s přidáním obvodů požadovaných pro dekódování hloubkové složky z videosignálu se zakódovaným třetím rozměrem pro vhodnou změnu chování vertikálního vychýlení rozmítaného elektronového paprsku pro vytvoření trojrozměrného efektu, obrázek 14 znázorňuje výhodné pří-

kladné provedení elektronických obvodů na bázi televize, které provádí funkci výběru hloubky a zobrazení, které jsou znázorněny na obr. 13 (b), obr. 15 znázorňuje alternativní optickou strukturu na úrovni obrazového bodu, v níž se poloha vstupního světla mění radiálně spíše než lineárně, obr. 16 je podobný obrázku 2, ale zobrazuje alternativní prostředky pro měnění vizuální vzdálenosti světla emitovaného z jednotlivého obrazového bodu od pozorovatele, obrázek 17 znázorňuje, jak je uspořádání znázorněného na obr. 16 dosaženo v praktickém příkladném provedení.

Příklady provedení vynálezu

Obrázek 1a znázorňuje ve značně zvětšeném tvaru jedno možné příkladné provedení optického prvku 2 použitého pro měnění vzdálenosti od pozorovatele, v níž se kolimovaný bod vstupu světla do tohoto přístroje může jevit. Pro jasnost výkladu, velikost takového optického prvku 2 se může značně měnit, ale je zamýšlena, aby byla přizpůsobena velikosti zobrazovaného obrazového bodu a jako taková bude pro televizní monitor typicky v řádu jednoho milimetru šířky a tří milimetrů výšky. Optiky tak malé jako 0,5 mm x 1,5 mm byly demonstrovány pro monitor počítače, který je konstruován pro pozorování z větší blízkosti, a tak velké jako 5 mm široké a 15 mm vysoké, což je velikost zamýšlená pro aplikaci ve velkorozměrových komerčních zobrazovačích, navržených pro pozorování ze značné vzdálenosti.

Materiály, z nichž byly tyto optiky na úrovni obrazového bodu vyrobeny, byly až doposud buď tavené křemíkové sklo s indexem lomu 1,498043 nebo jeden ze dvou plastů, a to po-

lymetalmetakrylát s indexem lomu 1,498 nebo metylmetakrylát s indexem lomu 1,558. Tímto se však nenaznačuje, že toto jsou jediné nebo ani výhodné optické materiály, z nichž mohou být takové optiky na úrovni obrazového bodu vyráběny.

Na obrázku 1 (a) je optický prvek na úrovni obrazového bodu pozorován šikmo zezadu, a jak je zřejmé, zatímco čelní plocha 1 tohoto optického přístroje je průběžně konvexní odshora až dolů, zadní povrch je tvarově proměnný postupně od konvexního nahoře ke konkávnímu dole. Úspěšně byly použity jak lineární, tak nelineární průběhy změn optických vlastností. Kolimovaný světelný paprsek je promítán přes optický přístroj ve směru optické osy 3, a jak je zřejmé, sběrné optické lomové plochy přístroje jimiž tento kolimovaný světelný paprsek prochází, se budou měnit s pohybem vstupního bodu paprsku odshora dolů na tomto přístroji.

Ačkoliv příkladné provedení zobrazené na obrázku 1 (a) obsahuje jednu pevnou plochu a jednu proměnnou plochu, jsou možné i variace tohoto návrhu, v nichž jsou proměnné obě plochy nebo v nichž jsou více než dvě optické lomové plochy. Obrázek 1(b) například znázorňuje druhé příkladné provedení, v němž optiky na úrovni obrazového bodu jsou složené optické přístroje sestávající ze tří optických prvků. Laboratorní testy naznačují, že složené přístroje optiky na úrovni obrazového bodu mohou zajišťovat zlepšenou kvalitu obrazu a zlepšený pozorovací úhel oproti optickým zařízením sestávajícím z jednoho prvku, a ve skutečnosti nejúspěšnější příkladné provedení této technologie dnes používá tříprvkovou optiku. Ovšem za účelem jasnosti výkladu se zde uvedené optické soustavy na úrovni optického prvku budou znázorňovat

jako soustavy sestávající z jediného prvku.

Obrázek 2 znázorňuje ve zhuštěné formě pro jasnost představy oči 4 pozorovatele v odstupu před optickým prvkem 2 na úrovni optického bodu. Kolimovaný světelný paprsek může vstoupit do zadní části optického přístroje 2 na různých bodech, z nichž tři jsou znázorněny jako světelné paprsky 5, 6, a 7. Poněvadž ohnisková vzdálenost přístroje 2 se mění v závislosti na vstupním bodě světelného paprsku, obrázek 2 znázorňuje, jak bude výsledný světelný bod představován pozorovateli na různých zdánlivých bodech 5a, 6a, nebo 7a v prostoru, odpovídajících jednotlivým dříve popsaným a očíslovaným místům vstupních paprsků. Ačkoliv body 5a, 6a, 7a jsou ve skutečnosti vertikálně vzdáleny jeden od druhého, tento vertikální posun není detekovatelný pozorovatelem, který vidí pouze zdánlivý posun v hloubce.

Obrázek 3 (a) znázorňuje, jak v jednom příkladném provedení tohoto vynálezu každý jednotlivý optický přístroj na úrovni obrazového bodu může být umístěn proti ploše obrazovky použité jako zobrazovací zdroj. V tomto obrázku optický prvek 2 leží na skleněném čelu 8 obrazovky, za nímž je běžná vrstva fosforů 9, která svítí, když na ní dopadne emitovaný a kolimovaný svazek elektronů znázorněný v různých polohách v tomto obrázku jako svazky 5b, 6b, 7b. Pro každou z těchto tří příkladných poloh elektronového svazku a pro každou další polohu elektronového svazku v prostorových mezích optického přístroje na úrovni optického bodu bude světelný bod vstupovat na jedinečném bodě zadní části optiky na úrovni obrazového bodu. Vertikální poloha elektronového svazku se může měnit za použití zcela běžných vychylovacích cívek

elektromagnetického svazku, jak se používají v běžných obrazovkách, podle specielně připraveného signálu, ačkoliv pokusy podniknuté v laboratoři naznačily, že zobrazování na vysoké četnosti rámců, to jest podstatně nad sto rámců za sekundu, mohou vyžadovat vychylovací cívky elektronového svazku, které jsou konstruovány tak, aby lépe reagovaly na vyšší vychylovací kmitočty, které jsou nezbytné při vysoké četnosti rámců. Vzorek fosforů na obrazovce však musí souhlasit s uspořádáním optik na úrovni obrazového bodu, a to jak v délce tak v prostorovém uspořádání, to jest optika musí být schopna být osvětlena pod ní ležícím fosforem v celé své navržené lineární vstupní ploše. Obrázek 3 (b) znázorňuje toto uspořádání přes šikmý zadní pohled na optiku 2 na úrovni optického bodu. V tomto schematu sousední fosforové obrazové body 35, z nichž devět je zobrazeno, budou mít tři různé barvy jako v běžné barevné obrazovce a budou mít v podstatě pravoúhlý tvar. Je třeba si všimnout, že velikost a poměr stran, to jest poměr délky a šířky každého fosforového obrazového bodu, v podstatě souhlasí s poměry stran na vstupním konci optiky na úrovni obrazového bodu, k níž je fosforový obrazový bod přivrácený. Jak je zřejmé při pozorování vystínovaného fosforového obrazového bodu, elektronový svazek rastrující přes fosforový obrazový bod může být zaostřen na jakémkoliv bodě podél délky fosforového obrazového bodu, znázorněného zde týmiž třemi příkladnými elektronovými svazky 5b, 6b, 7b. Výsledkem je, že bod, na němž se emituje světlo, je mírně posunut v rozmezí tohoto obrazového bodu.

Obrázek 3 (c) znázorňuje důležitost velikosti a poměru stran světelného paprsku, který vstupuje do optického pří-

stoje 2 na úrovni optického prvku, který je zde znázorněn zezadu. Vizuální zobrazení hloubky v televizní obrazovce je v požadavcích na rozlišení podobnější zobrazení chrominance či barvy než zobrazení luminance, to jest jasu, či černobílé složky videozobrazení. Tímto je míněno, že většina z přijímaného jemného detailu ve videozobrazení je přenášena jasovou složkou obrazu s relativně vysokým rozlišením, nad níž je zobrazena chrominanční složka s nižším rozlišením. Je možné mít mnohem nižší rozlišení v chrominanci, poněvadž oko odpouští mnohem více pokud jde o příjem barvy než pokud jde o příjem detailu obrazu. Výzkum v laboratoři naznačil, že oko podobně odpouští pokud jde o příjem hloubky v televizním obrazu.

Poté, co bylo toto řečeno, je však zobrazení pozorovatelné hloubky stále generováno fyzikálním pohybem světelného paprksu, který vstupuje do lineárního optického přístroje na úrovni optického bodu, a bude zřejmé, že čím větší je rozsah pohybu tohoto vstupního světelného paprsku, tim větší bude příležitost ovlivnit pozorovatelnou hloubku.

Na obrázku 3 (c) optický přístroj 2 na úrovni optického prvku je přibližně 3x tak vysoký jak široký. Kolimovaný vstupní světelný paprsek 66a, který je zde znázorněn v řezu, je kruhový a má průměr aproximující šířku optického přístroje 2. Kolimovaný vstupní světelný paprsek 66b je rovněž kruhový ale má průměr přibližně jednu pětinu délky optického přístroje 2. Toto na jedné straně umožňuje kolimovanému vstupnímu světelnému paprsku větší rozsah pohybu než je tomu u kolimovaného vstupního světelného paprsku 66a pro zajištění většího rozsahu pozorovatelné hloubky ve výsledném obra-

ze, ale na druhé straně toto je na úkor plochy průřezu osvětlujícího paprsku, která je pouze 36% plochy paprsku 66a. Pro udržení srovnatelného jasu ve výsledném obrazu by intenzita vstupního paprsku 66b musela být asi 2,7x vyšší než intenzita vstupního paprku 66a, což je nárůst, který je zcela dosažitelný.

Světelný paprsek 66c je stejně široký jako optický přístroj na úrovni optického prvku, ale je to horizontální ovál o výšce světelného paprsku 66b, to jest pouze jedna pětina výšky optického přístroje 2. Výsledný průřez oválu osvětlovacího paprsku je méně jasný než kruhový paprsek 66a, ale téměř 2x tak jasný než menší kruhový paprsek 66b. Tento návrh je vysoce funkční a zaostává pouze za perfektně pravoúhlým průřezem světelného paprsku 66d. To je ve skutečnosti průžez paprsku použitého v nejnovějším a nejvýhodnějším příkladném provedení tohoto vynálezu

Obrázek 4(a) znázorňuje, jak jsou optiky 2 na úrovni obrazového bodu uspořádány do soustavy řad, z nichž dvanáct je zobrazeno za účelem ilustrace, a jak jsou tyto umístěny na čele zdroje osvětlení zde zobrazeného jako obrazovka 10 v jednom příkladném provedení. Jak je řízený elektronový svazek rastrován přes řadu optik na úrovni obrazového bodu, jeho vertikální posuv se mění individuálně pro každý obrazový bod a vytváří tak horizontální rastrovací řádek, který je za účelem ilustrace znázorněn jako řádek 15, zobrazený jednak čárkovaně za soustavou obrazových bodů a odděleně pro jasnost pevnou čárou uvnitř elipsy nalevo. Jak je zřejmé, horizontální rastrovací řádek, který je v běžných zobrazeních obrazovky přímý, je pro každý jednotlivý obrazový bod

mírně posunut od středové čáry rastru, čímž se vytváří obraz, který, měníce svou vzdálenost od pozorovatele jak je to provedeno obrazový bod od obrazového bodu, obsahuje podstatné rozlišení v příjmu vjemu hloubky.

Zkušenosti ukázaly, že malá vmezeřená štěrbina mezi jednotlivými optickými prvky na úrovni optického bodu minimalizuje vzájemné pronikání signálů mezi optickými prvky, což má za následek zvýšenou jasnost obrazu a že tato izolace optik může být dále zvýrazněna vložením černého neprůhledného materiálu do těchto mezilehlých prostorů. Vmezeřené štěrbiny řádu 0,25 mm se ukázaly být zcela úspěšnými, ale byly demonstrovány i štěrbiny o šířce 0,10 mm a i tyto štěrbiny fungovaly perfektně jako optické izolátory, zejména když byly vyplněny výše zmíněným neprůhledným materiálem.

Soustavy těchto optik na úrovni optického bodu byly vytvořeny v procesu ručního připojováíní každé individuální optiky k povrchu vhodné obrazovky za použití opticky neutrálního tmelu. Tento proces je samozřejmě pracný a je náchylný k chybám umístění vzhledem k omezením přesnosti ruční mechaniky. Soustavy optik však byly velmi úspěšně vyráběny procesem vytváření kovových předloh kompletních soustav optik v negativu a pak vylisováním použitelných soustav optik do termoplastických materiálů pro vytvoření lisované repliky předlohy, která je pak přilepena ve své celosti k povrchu obrazovky. Provádění replik s velmi jemnými detaily na povrchu vylisováním bylo pozvednuto na vysokou úroveň v posledních letech vzhledem k technickým požadavkům na vytváření replik médií s velmi jemnými detaily a bohatými na informace, jako jsou laserové disky a kompaktní disky, což

jsou média, z nichž se s vysokou přesností a za nízké náklady vytvářejí repliky do laciných plastických materiálů. Předpokládá se, že výhodné výrobní techniky pro generování masově vyráběných soustav optik na úrovni optického prvku budou i nadále procesem vylisování do termoplastických materiálů. Vynálezci rovněž úspěšně vytvořili v laboratoři soustavy optik na úrovni obrazového bodu technikou injekčního lití. Doposud byly tři vrstvy různých optik na úrovni optického bodu, z nichž každá představuje optický prvek, úspěšně scentrovány pro vytvoření soustavy tříprvkových mikrooptik. V některých výhodných provedeních jsou tyto vrstvy slepeny, aby se pomohlo udržet scentrování, ale v jiných jsou upevněny na svých okrajích a nejsou vzájemně slepeny.

Při umístění optik na úrovni obrazových bodů na plochu obrazovky nebo jiného světlo emitujícího přístroje je přesné scentrování optik s podložními obrazovými body kritické. Vertikální rozcentrování způsobuje, že výsledný obraz má permanentní zkreslení v zobrazované hloubce, zatímco horizontální rozcentrování způsobuje omezení příčného pozorovacího rozsahu umožňovaného třídimenzionálním zobrazovacím přístrojem. Optické napojení mezi světlo generujícími optickými body a vstupní plochou optik na úrovni obrazového bodu je také zvýrazněna minimalizováním fyzické vzdálenosti mezi osvětlujícím fosforem a vstupní plochou optik tam, kde je to možné. V prostředí obrazovky toto implikuje, že sklo čelní plochy obrazovky, k níž jsou optiky přiloženy, by mělo být o minimální tloušťce, kterou je možno si dovolit při odpovídající strukturální integritě. Ve velkých obrazovkových monitorech může být tato čelní plocha tlustá až 8 mm, ale bylo

úspěšně ukázáno použití těchto optik se speciálně konstruovanou obrazovkou s tloušťkou čelní stěny 2 mm. Bylo zkonstruováno vysoce úspěšné příkladné provedení obrazovky, v němž byly optiky na úrovni obrazových bodů ve skutečnosti vytvořeny z čelní plochy obrazovky.

Obrázky 3 (b) a 4 (a) znázorňují v podstatě pravoúhlý vzorek obrazových bodů 35 zobrazovací obrazovky a lineární optické prvky 2 na úrovni obrazových prvků, to jest matici, v níž jsou řady přímé a scentrované obrazový bod za obrazovým bodem, a to s řadami jak nahoře, tak dole. Tento vzorek obrazových bodů a optik vytváří vysoce přijatelný trojrozměrný obraz, ale nelze předpokládat, že je to jediný takový vzorek, který je v rozsahu vynálezu možný.

Obrázek 4 (b) znázorňuje druhý příkladný vzorek obrazových bodů 35, v němž horizontální skupiny tří obrazových bodů jsou vertikálně posunuty vůči těm, které jsou nalevo a napravo od této skupiny, čímž se vytváří taškový vzorek skupin trojic obrazových bodů. Poněvadž tato konfigurace byla vytvořena v laboratoři, skupiny trojic obrazových bodů obsahují vždy jeden červený obrazový bod 35r, jeden zelený obrazový bod 35g a jeden modrý obrazový bod 35b. Tak jako v běžné dvourozměrné televizní obrazovce, barevné obrazy se vytvářejí z relativního osvětlení skupin nebo triád obrazových bodů z těchže samých tří barev. Odlišné uspořádání tří barev je v každé triádě možné, ale uspořádání zobrazené na obrázku 4(b) je příkladné provedení, které bylo doposud v laboratoři vytvořeno.

Obrázek 5 znázorňuje malou modifikaci signálu hloubky horizontálních rastrovacích čar v rastrovaném obrazu jako je

běžný televizní obraz. V běžné televizní obrazovce nebo monitoru počítače znázorněném vpravo nahoře na obr. 5, každý individuální obraz v pohybové sekvenci je vytvářen elektronovým svazkem rastrujícím horizontálně řádek za řádkem směrem dolů na stínítku znázorněném na obrázku 5 čtyřmi representativními rastrovými čarami 17. Toto vysoce pravidelné rastrování je řízeno elektronikou televizního přístroje nebo počítačového monitoru, a to generátorem 16 horizontálního rastrovacího řádku, a ani variace v luminanční nebo barvonosné složce signálu nevytvářejí variace v pravidelném postupu horizontálních rastrovacích řádků shora dolů.

Tento vynález přidává k tomuto pravidelnému postupu variaci ve formě malého posuvu od přímého horizontálního rastru, který vytváří hloubkový účinek. Taková variace je fyzikálně uskutečněna použitím generátoru 18 signálu hloubky, kde signál hloubky se přidává prostřednictvím sčítačky 19 k přímému horizontálnímu řádku pro vytváření malých variací ve vertikální poloze každého horizontálního rastrovacího řádku, čímž se vytvářejí řádky, které representativně připomínají řádky 20. Generátor signálu hloubky, znázorněný na obr. 5, je generické funkční vyjádření. V televizním přístroji je generátor signálu hloubky běžný dekodér obrazového signálu, který v současnosti vybírá luminanční, barvonosnou a časovací informaci z přijatého obrazového signálu a který je nyní doplněn jak je popsáno níže pro vybírání informací o hloubce, která byla zakódována do tohoto signálu zcela analogickým způsobem. Podobně v počítači je generátor hloubkové složky softwérem řízená obrazová karta, jako je obrazová karta VGA, která v současné době zajišťuje luminanční,

barvonosnou a časovací informaci pro monitor počítače a která rovněž zajistí programem řízenou informaci o hloubce pro tento monitor.

Obrázek 6 znázorňuje způsob, kterým lze použít zezadu osvětlený obraz z filmu 14 pro zajištění řízeného vstupního osvětlení pro optický přístroj na úrovni optického bodu v dalším výhodném příkladném provedení tohoto vynálezu. V tomto příkladu část filmu, která je umístěna za zobrazeným optickým prvkem je neprůhledná s výjimkou jediného průhledného bodu, vytvořeného pro umožnění vstupu světla do optického přístroje v požadovaném bodě. Filmový pásek je běžně osvětlen zezadu, ale pouze světelnému paprsku 5c je umožněno, aby průhledným bodem ve filmu prošel optickým prvkem 2. Jak je zřejmé, tato situace je analogická situaci na obr. 3, na níž řízený elektronový svazek v obrazovce byl použit pro výběr umístění osvětlovacího svazku. Použité zezadu osvětlené obrazy na filmu mohou být libovolné velikosti a byla postavena příkladná provedení používající průsvitných bodů velikých až 8 palců x 10 palců.

Obrázek 7 znázorňuje způsob, jakým může být soustava 11 optických prvků 2 na úrovni optického bodu, z nichž dvanáct je zobrazeno za účelem ilustrace, použito pro zobrazení obrazů ze specielně připraveného filmového pásu 13. Optická soustava 11 je na svém místě držena držákem 12. Obraz na filmovém pásu 13 je běžným způsobem zezadu osvětlen a výsledný obraz zaostřen běžným systémem projekčních čoček, zde představovaným čárkovaným kruhem 22, na soustavu 11, která je koaxiální s filmovým pásem 13, a projekční čočkou 22 na optické ose 23. Generovaný trojrozměrný obraz může být pozo-

rován přímo nebo může být použit jako generátor obrazu pro projektor trojrozměrného reálného obrazu známého typu. Generované trojrozměrné obrazy mohou být pozorovány jako nehybné obrazy nebo ve sledu jako skutečné trojrozměrné pohyblivé obrazy s toutéž četností rámců jako běžné filmy. V tomto příkladném provedení mohou být jednotlivé obrazové body ve filmovém pásu 13 značně menší než ty, které se používají pro televizní zobrazení, poněvadž výsledné obrazové body jsou zamýšleny pro zvětšení při projekci, přičemž výhoda rozlišení fotografického filmu vůči televiznímu zobrazení snadno stráví toto zmenšení velikosti obrazového bodu.

Obrázek 8 znázorňuje scénu v níž se používá dvou kamer pro určení hloubky každého objektu ve scéně, to jest vzdálenosti kteréhokoliv objektu ve scéně od hlavní zobrazovací kamery. Snímaná scéna zde v pohledu shora je představována pevným obdélníkem 24, pevným čtvercem 25 a pevnou elipsou 26, každý v různé vzdálenosti od hlavní zobrazovací kamery 27 a tím každý mající odlišnou hloubku ve snímané scéně. Hlavní zobrazovací kamera se používá ke snímání scény v jejím základním detailu z umělecky preferovaného směru. Sekundární kamera 28 je umístěna v určité vzdálenosti od první kamery a snímá scénu zešikma a tím zachycuje odlišný pohled na tutéž scénu současně s hlavní zobrazovací kamerou. Dobře známé techniky geometrické triangulace mohou být pak použity pro určení skutečné vzdálenosti od hlavní zobrazovací kamery, v níž je každý objekt scény.

Výhodným způsobem, kterým takové výpočty mohou být prováděny a výsledný signál hloubky generován, je v postprodukčním stupni, v němž výpočty týkající se generování

signálu hloubky jsou prováděny off-line, to jest po uskutečnění snímání obrazu a obecně na místě vzdáleném od místa snímání obrazu a při rychlosti generování signálu hloubky, která nemusí být vztažena k rychlosti snímání obrazu v reálném čase. Druhým preferovaným způsobem generování signálu hloubky je provádění tohoto potřebného výpočtu v reálném čase, to jest v podstatě při snímání obrazu. Výhodou generování signálu hloubky v reálném čase je, že umožňuje produkci živého trojrozměrného zobrazení.. Výpočetní požadavky na výrobu v reálném čase jsou však podstatně větší než požadavky na proces off-line, v němž rychlost může být snížena pro možnost použití nižší výpočetní schopnosti za nižší náklady. Pokusy prováděné v laboratoři naznačují, že pro provádění požadovaných výpočtů v reálném čase, které je výhodné z důvodů ceny a kompaktnosti elektronického designu, lze použít číslicových signálních procesorů určených pro zpracování obrazu, to jest čílicových obrazových procesorů, z nichž oba jsou specializované procesory s úzkou funkcí, ale s vysokou rychlostí.

Poněvadž sekundární kamera 28 se používá pouze pro zachycení objektů z úhlu odlišného od úhlu hlavní zobrazovací kamery, tato sekundární kamera může mít poněkud nižší zobrazovací kvalitu než hlavní zobrazovací kamera a tím může být i levnější. Zejména při aplikacích s pohyblivými obrazy, zatímco hlavní zobrazovací kamera bude nákladná a bude používat drahý film, sekundární kamera může být pevná kamera filmového typu nebo videokamera. Proto v kontrastu k běžným technikám stereoskopického filmování v nichž je třeba používat dvě kamery, z nichž každá používá nákladný 35 mm nebo

70 mm film, poněvadž každá z nich je hlavní zobrazovací kamerou, technika podle vynálezu vyžaduje použití pouze jedné vysoce kvalitní nákladné kamery, poněvadž je zde pouze jedna hlavní zobrazovací kamera.

Zatímco tato komparativní analýza dvou obrazů téže scény získaných z různých úhlů se ukázala být nejúspěšnější, je také možné získat poznatky o hloubce ve scéně použitím čelně umístěných aktivních nebo pasivních čidel, které nemusejí být inherentně zobrazovací čidla. V laboratoři bylo úspěšně dosaženo kompletního přiřazení hloubky celé schény, obrazový bod po obrazovém bodu, které bylo v laboratoři pojmenováno jako hloubková mapa, použitím soustavy komerčně dostupných ultrazvukových detektorů pro zjištění odraženého ultrazvukového záření, které bylo použito pro osvětlení scény. Podobně byl úspěšně použit rastrující infračervený detektor pro postupné snímání odraženého infračerveného záření, které bylo použito pro osvětlení scény. Konečně v laboratoři byly uskutečněny úspěšné pokusy s použitím mikrovlnného záření jako osvětlovacího zdroje a s mikrovlnnými detektory pro snímání odraženého záření. Tato technika může být zejména užitečná pro snímání trojrozměrných obrazů za použití radarového systému.

Obrázek 9 (a) znázorňuje základní kroky v procesu, jímž mohou být pro běžné dvourozměrné zobrazení odvozeny signály hloubky, čímž se umožní proces zpětného nastavení trojrozměrného obrazu na běžné dvojrozměrné zobrazování jak filmu, tak videa.

Na obrázku 9 (a) jsou tytéž řady tří předmětů 24, 25, a 26, které byly znázorněny v pohledu shora na obrázku 8,

nyní pozorovány na monitoru zepředu. Ve dvourozměrném monitoru 29 není samozřejmě zřejmý rozdíl v hloubce pro pozorovatele.

V procesu přidání hloubkové složky k zobrazování dvourozměrných obrazů podle vynálezu je scéna nejdříve digitalizována v počítačové pracovní stanici za použití obrazové digitalizační desky. Kombinace softwéru definice předmětu, použití dobře známých technik detekce hrany a jiných technik pak definuje každý jednotlivý předmět ve scéně tak, že s každým předmětem může být zacházeno individuálně za účelem dodatečného přiřazení hloubky. Kde softwér není schopen adekvátně definovat a separovat objekty automaticky, lidský editor provádí vyjasnění podle svého úsudku za použití myši, světelného pera, dotykového stínítka a ukazovátka nebo podobných ukazovacích zařízení pro vyjasnění a definování předmětu. Jakmile je scéna oddělena do individuálních předmětů, lidský editor podle svého úsudku postupně definuje pro softwér relativní vzdálenosti od kamery, to jest zdánlivou hloubku každého předmětu ve scéně. Tento proces zcela závisí na úsudku editora a bude zřejmé, že špatný úsudek bude mít za následek produkci zkreslených trojrozměrných scén.

V následujícím kroku v procesu softwér postupně rastruje všechny obrazové body uvnitř scény a přiřazuje hloubkovou složku každému jednotlivému obrazovému bodu. Výsledkem procesu je složka hloubky na rastrovacím řádku 31 na monitoru 30, která představuje representativní hloubkový signál, který lze získat z řádky obrazových bodů přes střed monitorované scény 29, křížící každý předmět na stínítku. Pohled shora na umístění těchto předmětů znázorněných na obrázku 8 bude

korelovat s relativní hloubkou zjevnou v rastrovacím řádku 31 představujícím hloubkovou složku na obr. 9 (a).

Vzájemné propojení a činnost zařízení, které může být použito pro přidání hloubky zobrazení videa podle tohoto procesu, je znázorněno na obrázku 9 (b). V tomto obrázku obraz zpracující počítačová pracovní stanice se zabudovaným obrazovým digitizérem 71 řídí vstupní videorekordér 72 a výstupní videorekordér 73 a přepínač 74 obrazových matric, kde řízení je znázorněno čárkovaně na obrázku 9(b), zatímco tok signálu je znázorněn pevnými čarami. Obrazový digitizér přijímá rámec obrazu ve vstupním videorekordéru přes maticový spínač na příkaz z pracovní stanice. Rámec je pak digitalizován a použije se proces definice předmětu popsaný na obr. 9(a) pro výslednou digitální scénu. Když se vypočítá signál hloubky pro tento rámec, tentýž rámec je vložen do obrazového generátoru 75 NTSC spolu s vypočítanou hloubkovou složkou, která je přidána k obrazovému rámci ve správném místě v obrazovém spektru generátorem NTSC. Výsledný hloubkově zakódovaný obrazový rámec je pak zapsán do výstupního videorekordéru 73 a proces začne znova pro následující rámec.

Několik důležitých bodů týkajících se tohoto procesu se objevilo v průběhu jeho vývoje v laboratoři. Prvním takovým bodem je, že poněvadž složka hloubky se přidává generátorem NTSC, který injektuje pouze hloubkovou složku bez změny jakýchkoliv dalších aspektů signálu, může být původní obrazová část signálu zapsána do výstupního videorekordéru bez nutnosti předchozí digitalizace obrazu. Toto pak zabraňuje vizuální degradaci způsobené digitalizací obrazu a jeho kon-

vertováním do analogového tvaru a jedinou takovou degradací ke které dojde bude degradace způsobená dvojím převodem při procesu kopírování video, což je degradace, která je minimalizována použitím vysílacího formátu "component video" analogových videorekordérů jako jsou přístroje M-II nebo Betacam. Je zřejmé, jak je dobře známo v průmyslu zobrazování, že při použití zcela digitálních záznamových zařízení, ať už založených na počítači nebo na magnetické pásce, nedojde vůbec k žádné degradaci při převádění obrazu.

Druhým takovým bodem je, že poněvadž proces se uskutečňuje rámec po rámci, jsou pro přidání hloubky požadovány video-rekordéry nebo jiná záznamová zařízení nazývaná rámcově přesná. Editor musí být v případě potřeby schopen na každý individuální rámec dosáhnout a nechat pak zpracovaný rámec zapsat na správné místo na výstupní pásce, a pouze přístroje konstruované pro přístup ke každému jednotlivému rámci, například podle časového kódu SMPTE, jsou vhodné pro takové použití.

Třetím takovým bodem je, že celý proces může být dán pod počítačové řízení a může být proto zpracováván nejpohodlněji z jediného počítačového stojanu spíše než z několika oddělených soustav ovladačů. Vzhledem k dostupnosti videorekordéru s počítačem řízenou úrovní vysílání a dalších záznamových zařízení jak analogových, tak digitálních, určité aspekty procesu přidávání hloubky mohou být poloautomatizovány použitím takových spojení počítače s videorekordérem, jako jsou na čas náročné automatické převíjení a předvíjení.

Čtvrtým takovým bodem je, že softwér může být pro zvýšení kvality přidání hloubky na úrovni mikrorysů vybaven ur-

čitými aspekty toho, o čem se obecně hovoří jako o umělé inteligenci nebo strojní inteligenci. V laboratoři byla například vyvinuta a je dále zlepšována technika přidávající větší realitu přidání hloubky lidským tvářím za použití topologie lidské tváře, to jest faktu, že nos vyčnívá dále než líce, které se svažují zpět k uším atd., kde každý rys má svou vlastní hloubkovou charakteristiku. Toto usnadní požadavky na mnoho vstupů editora při jednání s mnoha běžnými objekty nacházejícími se ve filmu a na videu, přičemž lidské tváře jsou zde použitým příkladem.

Pátým takovým bodem je, že řídící softwér může být konstruován tak, aby pracoval poloautomatickým způsobem. Tím je míněno to, že pokud předměty ve scéně zůstáváají relativně konstantní, řídicí pracovní stanice může zpracovávat následné rámce automaticky a bez přídavných vstupů od editora, čímž se pomáhá zjednodušení a zrychlení procesu. Proces bude samozřejmě znovu vyžadovat vstup editora, pokud na scénu vstoupí nový objekt nebo pokud se perspektiva scény nadměrně změní. V laboratoři byly vyvinuty a v současné době se vylepšují techniky založené na umělé inteligenci, která automaticky vypočítává změny hloubky pro individuální předměty ve scéně, založené na změnách v perspektivě a relativní velikosti objektu na aspekty, které jsou softwéru známy.

Šestým takovým bodem je, že při práci s nehybným nebo pohyblivým filmem jako vstupním a výstupním médiem mohou být vstupní videorekordér 72, výstupní videorekordér 74 a přepínač 74 obrazové matrice případně nahraženy filmovým scannerem o vysokém rozlišení, číslicovým datovým přepínačem a filmovou tiskárnou a vysokém rozlišení. Zbytek procesu

zůstává v podstatě tentýž jako u výše popsané situace zpracování obrazu. Za těchto okolností se vkládání signálu hloubky za použití generátoru NTSC obchází filmovým procesem znázorným na obr. 8.

Sedmým takovým bodem je, že při práci v celodigitálním záznamovém prostředí, jako v ukládání obrazu na bázi počítače, jsou výstupní videorekordér 73, vstupní videorekordér 72 a přepínač 74 obrazové matrice účinně nahraženy velkou pamětí počítače. Taková velká paměť je typicky magnetický disk, jak je tomu v editující pracovní stanici na bázi počítače, která se používá v laboratoři, ale stejně tak to může být nějaká jiná forma velké paměti. V tomto celodigitálním prostředí se vkládání signálu hloubky nahrazuje přidáním prvků hloubkové mapy na úrovni obrazových bodů k běžnému formátu obrazu uloženého v počítači.

Jako doplněk A je připojena kopie softwérového programu použitého v laboratorních podmínkách pro dosažení dodatečného přizpůsobení diskutovaného výše s ohledem na obrázky 9 (a) a 9 (b).

Obrázek 10 znázorňuje aplikaci technik zobrazení hloubky na úrovni obrazového bodu, odvozených v průběhu tohoto vývoje, na trojrozměrné zobrazení tištěných obrazů. Scéna 32 je běžná dvojrozměrná fotografie nebo tištěná scéna. Matrice 33 mikročoček na úrovni obrazového bodu, která je zde pro jasnost znázorněna přehnaně, je přiložena na dvourozměrný obraz tak, že každá mikročočka má odlišnou ohniskovou délku a proto představuje tento obrazový bod na odlišné zdánlivé hloubce pro oko pozorovatele. Při velmi zvětšeném pozorování v průřezu 34 každá mikročočka může být viděna ja-

ko tvarově specifická a proto specifická z hlediska optické charakteristiky pro zajištění vhodného příjmu hloubky daného obrazového bodu pozorovatelem. Zatímco v laboratoři byly doposud používány mikročočky s průměry tak malými jako jeden milimetr, byly prováděny pokusy s mikročočkami o rozměrech ve zlomcích milimetru, což dovozuje, že soustavy čoček těchto velikostí jsou zcela možné a že budou mít za následek trojrozměrné tištěné zobrazení s vynikajícím rozlišením.

Při hromadné výrobě se předpokládá, že zde popsané techniky generování signálu hloubky budou použity pro výrobu tištěných matric, z nichž mohou být opět vylisovány do lisovatelného nebo termoplastického materiálu levné soustavy mikročoček pro daný obraz způsobem analogickým lisování data nesoucích ploch kompaktních disků nebo hromadně či opakovaně vyráběných odrazových hologramů, typicky prováděných na kreditních kartách. Takové techniky slibují velkovýrobní levné třírozměrné tištěné zobrazení pro zahrnutí do časopisů, novin a dalších tištěných médií. Zatímco matrice 33 mikročoček je zobrazena v pravoúhlém vzoru, i další vzory jako koncentrické kruhy mikročoček se zdají fungovat docela dobře.

Je důležité poznamenat, že obrazová nebo luminanční nosná v běžném obrazovém signálu NTSC zaujímá značně větší šířku obrazového pásma než barvonosný signál nebo signál hloubky nesoucí pomocné nosné. Luminanční složka obrazu NTSC má relativně vysoké rozlišení a je často charakterizována jako obraz kreslený jemnou tužkou. Barvonosný signál na druhé straně musí přenášet značně méně informací pro vytvoření akceptovatelného barevného obrazu v televizním obrazu a je často charakterizován jako široký štětec malující směs barev

na černobílý obraz o vysokém rozlišení. Signál hloubky v tomto vynálezu je ve svém stylu podobnější signálu barvy ve svém omezeném požadovaném obsahu informací než obrazové nosné s vysokým rozlišením.

Jedním z kritických bodů řízení obrazového signálu je, jak zakódovat do signálu informaci, která nebyla přítomna, když byl konstruován originál, a jak to udělat, aniž by se zmátla nebo jinak učinila zastaralou instalovaná základna televizních přijímačů. Obrázek 11 znázorňuje distribuci energie běžného obrazového signálu NTSC, znázorňující obrazovou nebo jasovou nosnou 36 a nosnou 37 chrominanční nebo barvonosné informace. Všechna informace v obrazovém spektru je nešena energií v oddělených kmitočtových intervalech, zde představovaných oddělenými vertikálními čarami. Zbytek spektra je prázdný a neužívaný. Jak je zřejmé z obrázku 11, tvůrci barevného obrazového signálu NTSC úspěšně zabudovali značné množství přídavné informace, to jest informace o barvě, do zavedené konstrukce signálu použitím téže koncepce koncentrování energie signálu na odlišných kmitočtových bodech a pak prokládáním těchto bodů mezi ustanovenými body energie obrazové nosné tak, že se tyto nepřekrývají a spolu neinterferují.

Podobným způsobem tento vynález kóduje další přídavnou informaci ve formě požadovaného signálu hloubky do existující struktury obrazového signálu NTSC za použití téhož prokládacího procesu, který se používá s barvonosným signálem. Obrázek 12 znázorňuje tento proces tak, že opět ukazuje tutéž nosnou 36 luminance a pomocnou nosnou 37 barvonosného signálu jako na obrázku 11 s přidáním pomocné nosné 38

hloubky. Pro informaci pomocná nosná barvonosného signálu zaujímá přibližně 1,5 MHz šířky pásma se středem na 3,579 MHz, zatím co pomocná nosná hloubky zaujímá pouze přibližně 0,4 MHz se středem na 2,379 MHz. Takto pomocné nosné barvonosného signálu a signálu hloubky, každá proložená s nosnou luminančního signálu, jsou dostatečně odděleny tak, aby spolu neinterferovaly. Zatímco uvedený kmitočet pomocné nosné a uvedená zaujímaná šířka pásma pracují docela dobře, jsou ve skutečnosti možné i jiné hodnoty. Například v pokusech prováděných v laboratoři bylo úspěšně demonstrováno podstatné zmenšení uvedeného pásma 0,4 MHz pro pomocnou nosnou hloubky použitím dobře známých zhušťovacích technik na signál hloubky před jeho vložením do signálu NTSC. Toto je následováno na straně příjmu signálu dekompresí a výběrem před použitím pro řízení hloubku zobrazujícího přístroje. Podobné přístupy pro usazení signálu hloubky do obrazových formátů PAL a SECAM byly rovněž testovány v laboratoři, ačkoliv specifika struktury a příslušné kmitočty se mění vzhledem k měnící se struktuře těchto obrazových signálů. V celodigitálním prostředí, jako v obrazové paměti na bázi počítače, existuje široké spektrum formátů uchování obrazu a proto způsob přidávání bitů určených pro uložení hloubkové mapy se mění od formátu k formátu.

Obrázek 13 (a) znázorňuje funkční tvar obvodů v běžném televizním přijímači, který typicky řídí vertikální vychýlení rastrovacího elektronového svazku v obrazovce za použití terminologie běžné v televizním průmyslu. Zatímco některé detaily se mohou měnit od druhu ke druhu a od modelu k modelu, podstatné rysy zůstávají stejné.

V tomto diagramu, představujícím běžný návrh televizního přijímače, je účelem generování rozmítání rastrovacího elektronového svazku, který je konzistentní a synchronizovaný s příchozím obrazovým signálem. Signál je přijímán tunerem 49 a zesílen obrazovým mezifrekvenčním zesilovačem 50, pak odeslán do obrazového detektoru 51 pro výběr obrazového signálu. Výstup obrazového detektoru 51 je zesílen ve výstupním zesilovači 52 detektoru, dále zesílen v prvním obrazovém zesilovači 53 a prochází zpožďovacím vedením 54.

V běžném obrazovém signálu jsou tři hlavní složky, jasová složka, to jest černobílá část signálu, barvonosná složka a časovací složka signálu, která se týká zajištění toho, že se všechno stane podle správně načasovaného plánu. Z těchto složek je synchronizační informace oddělena od zesíleného signálu v synchronizačním separátoru 55 a vertikální synchronizační informace je pak invertována ve vertikálním synchronizačním invertoru 56 a přivedena do generátoru 64 vertikálního rozmítání. Výstup tohoto generátoru rozmítání je přiveden do elektromagnetické cívky v obrazovce známé jako vychylovací jho 65. Toto vychylovací jho způsobuje rozmítání elektronového svazku, který sleduje hladkou a přímou dráhu při protínání stínítka obrazovky.

Jak bylo popsáno dříve, v televizní obrazovce pro trojrozměrný obraz jsou do této přímé dráhy elektronového svazku vnášeny malé variace, které prostřednictvím optik na úrovni obrazového bodu vytvářejí trojrozměrný účinek. Obrázek 13 (b) znázorňuje v téže funkční formě přídavné obvody, které musí být přidány k běžné televizi, aby vybraly hloubkovou složku vhodně zakódovaného obrazového signálu a přeložily

tuto hloubkovou složku signálu do jemně proměnné dráhy rastrovacího elektronového svazku. V tomto schematu jsou funkce vně přerušované čáry funkcemi běžného televizního přijímače, jak je znázorněn na obrázku 13(a), a ty, které jsou uvnitř této přerušované čáry představují přídavky požadované pro vyjímání hloubkové složky a generování trojrozměrného účinku.

Jak bylo posáno na obr. 12, signál hloubky je zakódován do obrazového signálu NTSC způsobem v podstatě identickým se způsobem zakódování barvonosného signálu, ale prostě na odlišném kmitočtu. Poněvadž kódovací proces je týž, může být signál obsahující hloubkovou složku zesilován na úroveň dostačující pro výběr za použití téhož zesilovače, který se používá v běžném televizním přijímači pro zesilování signálu barvy před jeho výběrem, který je zde označen jako první mezifrekvenční zesilovač 57 barvy.

Tato zesílená hloubková složka signálu se vybírá z obrazového signálu v procesu identickém s procesem používaným pro výběr zakódované barvy v témže signálu. V tomto procesu je referenční signál generován televizním přijímačem na kmitočtu na němž by měl být signál hloubky. Tento signál se porovnává se signálem, který je skutečně přítomen na tomto kmitočtu a jakékoliv rozdíly od referenčího signálu se považují za signál hloubky. Tento referenční signál je generován tvarovačem 59 hradlového impulsu hloubky a tvarován na požadovanou úroveň omezovačem 58 hradlového impulsu hloubky. Plně vytvořený referenční signál je synchronizován s příchozím zakódovaným signálem hloubky pro tentýž synchronizační separátor 55, používaný pro synchronizaci horizontálního rozmí-

tání elektronového svazku v běžném televizním přijímači.

Když zesílený zakódovaný signál hloubky z prvního mezifrekvenčního zesilovače 57 barvy a referenční signál z omezovače 58 hradlového impulsu hloubky se smísí pro srovnání, jsou výsledky zesíleny s hradlovým synchronizačním zesilovačem 63 hloubky. Tento zesílený signál bude obsahovat jak složku barvy, tak složku hloubky, takže pouze ty signály, které se nacházejí v okolí kmitočtu 2,379 MHz kódovacího kmitočtu hloubkového signálu, jsou vybrány extraktorem 62. Toto je pak vybraný signál hloubky, který je následně zesílen na užitečnou hladinu výstupním zesilovačem 61 extraktoru.

Po výběru hloubkové složky z kompozitního obrazového signálu obvod musí modifikovat hladké horizontální rozmítání elektronového svazku přes televizní stínítko pro umožnění zobrazení hloubky ve výsledném obrazu. Pro modifikaci tohoto horizontálního rozmítání, vybraný a zesílený signál hloubky se přidá ve sčítačce 60 hloubky ke standartnímu vertikálnímu synchronizačnímu signálu běžně generovanému v běžném televizním přijímači, jak bylo popsáno dříve na obr. 13(a). Modifikovaný vertikálni synchronizační signál, který je výstupem ze sčítačky 60 hloubky, se nyní používá pro vytváření vertikálního rozmítání elektronového svazku v generátoru 64 vertikálního rozmítání, který jako v běžném přijímači řídí vychylovací jho 65, které řídí pohyb rozmítaného elektronového svazku. Konečným výsledkem je rozmítaný elektronový svazek, který je nepatrně vychylován nahoru nebo dolů od své běžné středové čáry pro generování trojrozměrného účinku v obraze nepatrnými změnami vstupního světelného bodu na

dříve popsané optiky na úrovni obrazového bodu.

Obrázek 14 znázorňuje elektronické obvody, které představují výhodné příkladné provedení přídavných funkcí popsaných uvnitř obdélníků z přerušované čáry z obr. 13.

Obrázek 15 znázorňuje alternativní prostředky pro změny polohy světla, které vstupuje do jinak vytvořené optické struktury na úrovni obrazového bodu. V této alternativě optická struktura 39 na úrovni obrazového bodu má vhodnou funkci optického přenosu, která zajišťuje ohniskovou délku, která se zvětšuje radiálně směrem ven od osy optického prvku 39 a je symetrická okolo své osy 43. Světlo, kolimované do válcového tvaru, vstupuje do optické struktury a poloměr válce kolimovaného světla se může měnit od 0 až do efektivního operačního poloměru optické struktury. Jsou znázorněny tři takové možné válcové kolimace 40, 41 a 42, vytvářející z čelního pohledu prstencové vstupní světelné pásy 40a, 41a, případně 42a, z nichž každý vytváří podle specifické optické přenosové funkce zařízení generovaný světelný obrazový bod na různé zdánlivé vzdálenosti od pozorovatele.

Obrázek 16 znázorňuje ve zhuštěné formě pro jasnost výkladu ještě další alternativní prostředek měnění optické vzdálenosti od pozorovatele světla emitovaného z jednotlivého obrazového bodu. V tomto zobrazení je pozorovatelovo oko 4 v odstupu před optikou na úrovni obrazového bodu. Kolimovaný světelný paprsek může dopadat na šikmo umístěné zrcadlo 76 v proměnných bodech, z nichž tři jsou znázorněny jako světelné paprsky 5, 6 a 7. Zrcadlo 76 odráží vstupní světelný paprsek na šikmý úsek konkávního zrcadla 77, které podle charakteristik vytváření obrazu konkávního zrcadla dává svě-

telný paprsek o proměnné vizuální vzdálenosti od pozorovatele 5a, 6a, 7a odpovídající jednotlivým dříve popsaným a číslovaným umístěním vstupních paprsků. Konkávní zrcadlo může mít zakřivení, které lze popsat jako různé kuželosečky, přičemž v laboratoři se úspěšně používaly jak parabolická, tak hyperbolická a sférická zakřivení. V tomto příkladném provedení experimentální výsledky naznačují, že lze použít jak planární, tak zakřivená zrcadla.

Obrázek 17 znázorňuje, jak v jednom výhodném provedení uspořádání znázorněného na obr. 16 kombinace na úrovni obrazového bodu planárního zrcadla 76 a konkávního zrcadla 77 jsou uspořádány proti povrchu obrazovky použité jako osvětlovací zdroj. Ve výkrese konkávní zrcadlo z jednoho obrazového bodu je kombinováno s planárním zrcadlem ze sousedního obrazového bodu, bezprostředně nad, pro vytvoření kombinovaného prvku 78, který je uložen proti skleněnému čelu 8 obrazovky, za nímž jsou běžné fosforové vrstvy 9, které září pro vytváření světla, když na ně dopadne emitovaný a kolimovaný svazek elektronů zobrazený v různých polohách tohoto obrázku jako svazky 5b, 6b a 7b. Pro každou z těchto tří příkladných poloh a pro jakoukoli jinou polohu svazku v prostorových mezích optického přístroje na úrovni obrazového bodu světelný bod vstupuje v jedinečném bodě do sestavy a bude proto představován pozorovateli na odpovídajícím jedinečném bodě. Tak jako je tomu u lomových příkladných provedení tohoto vynálezu, je možno docela vhodně použít i jiné světelné zdroje než obrazovky.

```
//      3D0105.cpp                      APPENDIX A

//      AGENTS OF CHANGE INC.
//      Advanced Technology 3-D Retrofitting Controller Software
//      Employing Touch Screen Graphical User Interface
//      V.01.05

//      Includes the following control elements:


#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <iostream.h>

#define MOUSE           0x33
#define BUT1PRESSED     1
#define BUT2PRESSED     2
#define TRUE            1
#define FALSE           0

void ActivMouse()
{
        //      activate mouse.
        _AX=32;
        geninterrupt(MOUSE);
}

int ResetMouse()
{
        //      mouse reset.
        _AX=0;
        geninterrupt(MOUSE);
        return(_AX);
}

void ShowMouse()
{
        //      turn on mouse cursor.
        _AX=1;
        geninterrupt(MOUSE);
}

void HideMouse()
{
        //      turn off mouse cursor.
        _AX=2;
        geninterrupt(MOUSE);
}

void ReadMouse(int *v, int *h, int *but)
{
        int temp;
        _AX=3;
        geninterrupt(MOUSE);

        //      which button pressed:  1=left, 2=right, 3=both.
        temp=_BX;
        *but=temp;
```

```
        //      horizontal coordinates.
        *h=_CX;

        //      vertical coordinates.
        *v=_DX;
}


class Button
//      this class creates screen buttons capable of being displayed raised
//      or depressed.  Labels displayed on the buttons change colour when
//      the button is depressed.
{
public:
        int button_centrex, button_centrey, button_width, button_height;
        int left,top,right,bottom, text_size, text_fields, lfont;
        char button_text1[40], button_text2[40];
        unsigned upattern;
//      button_centrex, button_centrey is the centre of the button placement.
//      button_width and button_height are the dimensions of the button in pixels.
//      button_text is the label on the button.
//      text_size is the text size for settextstyle().

        int mouseX, mouseY, mouseButton;
        int oldMouseX, oldMouseY;
        int button1Down,button2Down;

        int pressed;

        Button(int x, int y, int width, int height, int tfields, char *btext1, char *btext2, int tsize, int f)
        //      this constructor initializes the button variables.
                {
                button_centrex = x;
                button_centrey = y;
                button_width = width;
                button_height = height;
                strcpy(button_text1, btext1);
                strcpy(button_text2, btext2);
                text_size=tsize;
                text_fields=tfields;
                lfont=f;

                left=button_centrex - button_width/2;
                top=button_centrey - button_height/2;
                right=button_centrex + button_width/2;
                bottom=button_centrey + button_height/2;

                oldMouseX=0; oldMouseY=0;
                button1Down=FALSE;
                button2Down=FALSE;

                pressed=FALSE;
                }

        void up()
        //      draws a raised button and prints the required label on it.
                {
                setcolor(5);
                setlinestyle(SOLID_LINE,upattern,NORM_WIDTH);
                setfillstyle(SOLID_FILL, LIGHTGRAY);
                bar3d(left,top,right,bottom,0,0);
                setcolor(WHITE);
                setlinestyle(SOLID_LINE,upattern,THICK_WIDTH);
```

```
                line(left+2,bottom-1,left+2,top+1);
                line(left+1,top+2,right-1,top+2);
                setcolor(DARKGRAY);
                setlinestyle(SOLID_LINE,upattern,NORM_WIDTH);
                line(left+4,bottom-3,right-1,bottom-3);
                line(left+3,bottom-2,right-1,bottom-2);
                line(left+2,bottom-1,right-1,bottom-1);
                line(right-3,bottom-1,right-3,top+4);
                line(right-2,bottom-1,right-2,top+3);
                line(right-1,bottom-1,right-1,top+2);
                //      put the required text in the button.
                setcolor(5);
                settextjustify(CENTER_TEXT, CENTER_TEXT);
                settextstyle(font, HORIZ_DIR, text_size);
//              cout << button_text2 << endl;
                if (text_fields==1)
                        outtextxy(button_centrex, button_centrey-4*(float(button_height)/50),
button_text1);
                        else
                        {
                        outtextxy(button_centrex, button_centrey-13*(float(button_height)/50),
button_text1);
                        outtextxy(button_centrex, button_centrey+9*(float(button_height)/50),
button_text2);
                        }

                pressed=FALSE;
                }

        void down()
        //      draw a depressed button and prints the required label on it.
                {
                setcolor(5);
                setlinestyle(SOLID_LINE,upattern,NORM_WIDTH);
                setfillstyle(SOLID_FILL, DARKGRAY);
                bar3d(left,top,right,bottom,0,0);
                setcolor(5);
                setlinestyle(SOLID_LINE,upattern,THICK_WIDTH);
                line(left+2,bottom-1,left+2,top+1);
                line(left+1,top+2,right-1,top+2);
                setcolor(LIGHTGRAY);
                setlinestyle(SOLID_LINE,upattern,NORM_WIDTH);
                line(left+4,bottom-3,right-1,bottom-3);
                line(left+3,bottom-2,right-1,bottom-2);
                line(left+2,bottom-1,right-1,bottom-1);
                line(right-3,bottom-1,right-3,top+4);
                line(right-2,bottom-1,right-2,top+3);
                line(right-1,bottom-1,right-1,top+2);
                //      put the required text in the button.
                setcolor(WHITE);
                settextjustify(CENTER_TEXT, CENTER_TEXT);
                settextstyle(font, HORIZ_DIR, text_size);
//          cout << button_text2 << endl;
                if (text_fields==1)
                        outtextxy(button_centrex, button_centrey-4*(float(button_height)/50.),
button_text1);
                        else
                        {
                        outtextxy(button_centrex, button_centrey-13*(float(button_height)/50.),
button_text1);
                        outtextxy(button_centrex, button_centrey+9*(float(button_height)/50.),
button_text2);
                        }
```

```
                pressed=TRUE;
                }

        int touched()
        //      determines whether a button has been touched, and returns
        //      TRUE for yes, and FALSE for no.  Touching is emulated
        //      by a mouse click.
                {
                int temp;
                _AX=3;
                geninterrupt(MOUSE);

        //      which button pressed:  1=left, 2=right, 3=both.
                temp=_BX;
                mouseButton=temp;

                //      horizontal coordinates.
                mouseX=_CX;

                //      vertical coordinates.
                mouseY=_DX;

                if (mouseButton&BUT1PRESSED)
                {
                        button1Down=TRUE;
                        return 0;
                }
                else if (button1Down)
                //      if button 1 was down and is now up, it was clicked!
                        {
                //      check whether the mouse is positioned in the button.
                        if ((((mouseX-left)*(mouseX-right))<0) && (((mouseY-top)*(mouseY-
bottom))<0))
                //      if this evaluates as TRUE then do the following.
                                {
                                button1Down=FALSE;
                                return 1;
                                }
                                button1Down=FALSE;
                                return 0;
                        }

                }

};


//      XXXXXXXXXXXXXXXXXX  M A I N  XXXXXXXXXXXXXXXXXXXX

void main()
{
//      this is the system main.

int Page_1_flag, Page_2_flag, Page_3_flag, Page_4_flag, Page_5_flag;
int Page_6_flag, Page_7_flag, Page_8_flag, Page_9_flag, Page_10_flag;
char which;

//      initialize the graphics system.
int gdriver = DETECT, gmode, errorcode;
initgraph(&gdriver, &gmode, "c:\\borlandc\\bgi");
//      read the result of initialization.
errorcode = graphresult();
```

```
if (errorcode != grOk) { // an error occurred.
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);
  }

//if (!ResetMouse())
//{
//      printf("No Mouse Driver");
//}

//      set the current colours and line style.
//      set BLACK (normally palette 0) to palette 5 (normally MAGENTA)
//      to correct a colour setting problem inate to C++.
setpalette(5, BLACK);

//      activate the mouse to emulate a touch screen.
//ActivMouse();
//ShowMouse();


//      construct and initialize buttons.
Button logo(getmaxx()/2,100,260,130,1,"(AOCI LOGO)","",4,1);
Button auto_control1(200,400,160,50,2,"AUTO","CONTROL",2,1);
Button manual_control1(400,400,160,50,2,"MANUAL","CONTROL",2,1);
Button mute1(568,440,110,50,1,"MUTE","",4,1);
//Button proceed(getmaxx()/2,440,160,50,1,"PROCEED","",4,1);
Button c_vision(getmaxx()/2,350,450,100,1,"3-D RETRO","",8,1);
Button main_menu(245,20,460,30,1,"M A I N  M E N U","",2,1);
Button time_date2(245,460,460,30,1,"Date:      Time:      Elapsed:      ","",2,1);
Button video_screen(245,217,460,345,1,"","",4,1);
Button video_message1(245,217,160,50,2,"Video Not","Detected",2,1);

Button auto_onoff2(555,20,130,30,2,"AUTO CONTROL","ON / OFF",5,2);
Button manual_control2(555,60,130,30,1,"MANUAL CONTROL","",5,2);
Button name_tags2(555,100,130,30,1,"OBJECT TAGS","",5,2);
Button voice_tags2(555,140,130,30,2,"TRIANGULATE/","DIST. CALC.",5,2);
Button custom_session2(555,180,130,30,1,"CUSTOM SESSION","",5,2);
Button memory_framing2(555,220,130,30,1,"MEMORY FRAMING","",5,2);
Button remote_commands2(555,260,130,30,2,"REMOTE ENDS","COMMANDS",5,2);
Button av_options2(555,300,130,30,2,"AUDIO/VISUAL","OPTIONS",5,2);
Button codec_control2(555,340,130,30,1,"CODEC CONTROL","",5,2);
Button mcu_control2(555,380,130,30,1,"MCU CONTROL","",5,2);
Button dial_connects2(555,420,130,30,2,"DIAL-UP","CONNECTIONS",5,2);
Button mute2(555,460,130,30,1,"MUTE","",5,2);
Button ind_id3(245,20,460,30,1,"PERSONAL IDENTIFICATION","",2,1);
Button frame_cam3(555,20,130,30,1,"FRAME CAMERA","",5,2);
Button cam_preset3(555,60,130,30,1,"CAMERA PRESET","",5,2);
Button autofollow3(555,180,130,30,1,"AUTOFOLLOWING","",5,2);
Button return3(555,420,130,30,2,"RETURN TO","LAST MENU",5,2);
Button touch_face3(130,418,230,35,2,"DEFINE AN OBJECT","AND THEN TOUCH:",5,2);
Button type_id3(308,418,105,35,2,"ACQUIRE","OBJECT",5,2);
Button write_id3(423,418,105,35,2,"LOSE","OBJECT",5,2);
Button cancel3(555,340,130,30,1,"CANCEL CHOICE","",5,2);
Button keyboard(245,375,450,200,1,"(Keyboard)","",2,1);
Button writing_space(245,425,450,100,1,"(Writing Space)","",2,1);
Button typing_done(555,260,130,30,2,"TYPE AND THEN","PRESS HERE",5,2);
Button writing_done(555,260,130,30,2,"WRITE AND THEN","PRESS HERE",5,2);
Button dial_connects6(getmaxx()/2,20,604,30,1,"DIAL-UP CONNECTIONS","",2,1);
Button directory6(getmaxx()/2,60,300,30,1,"DIRECTORY","",2,1);
Button manual_dialing6(57,420,84,30,2,"MANUAL","DIALING",5,2);
Button line_16(151,420,84,30,1,"LINE 1","",5,2);
```

```
Button line_26(245,420,84,30,1,"LINE 2","",5,2);
Button dial_tone6(339,420,84,30,1,"DIAL TONE","",5,2);
Button hang_up6(433,420,84,30,1,"HANG UP","",5,2);
Button scroll_up6(104,260,178,30,1,"SCROLL DIRECTORY UP","",5,2);
Button scroll_down6(292,260,178,30,1,"SCROLL DIRECTORY DOWN","",5,2);
Button dial_this6(198,300,84,30,2,"DIAL THIS","NUMBER",5,2);
Button add_entry6(104,340,178,30,1,"ADD AN ENTRY","",5,2);
Button delete_entry6(292,340,178,30,1,"DELETE AN ENTRY","",5,2);
Button keypad6(505,320,230,151,1,"(Keypad)","",2,1);


Page_1:
//      this is the opening screen.

//      set the current fill style and draw the background.
        setfillstyle(INTERLEAVE_FILL,DARKGRAY);
        bar3d(0,0,getmaxx(),getmaxy(),0,0);

        logo.up();
        c_vision.up();
//      proceed.up();
//      auto_control1.up();
//      manual_control1.up();
        mute1.up();
        settextstyle(TRIPLEX_FONT, HORIZ_DIR, 2);
        outtextxy(getmaxx()/2,190,"(C) 1993-1995 AGENTS OF CHANGE INC.");
        settextstyle(TRIPLEX_FONT, HORIZ_DIR, 4);
        outtextxy(getmaxx()/2,235,"WELCOME");
        outtextxy(getmaxx()/2,265,"TO");
        Page_1_flag=TRUE;

while (Page_1_flag)
{
//      temporary keypad substitute for the touch screen.
which = getch();

        if (which=='1')
        {
                if (!c_vision.pressed)
                {
                        c_vision.down();
                        goto Page_2;
                }
                else c_vision.up();
        }

        if (which=='2')

        {
                if (!mute1.pressed) mute1.down();
                else mute1.up();
        }


if (which=='S') Page_1_flag=FALSE;
}
goto pgm_terminate;

Page_2:
//      this is the main menu.

        setfillstyle(INTERLEAVE_FILL,DARKGRAY);
```

```
        bar3d(0,0,getmaxx(),getmaxy(),0,0);
        main_menu.up();
        video_screen.up();
        video_message1.down();
        time_date2.up();
        auto_onoff2.up();
        manual_control2.up();
        name_tags2.up();
        voice_tags2.up();
        custom_session2.up();
        memory_framing2.up();
        remote_commands2.up();
        av_options2.up();
        codec_control2.up();
        mcu_control2.up();
        dial_connects2.up();
        mute2.up();

        Page_2_flag=TRUE;

while (Page_2_flag)
{
//      temporary keypad substitute for the touch screen.
which = getch();

        if (which=='1')
        {
                if (!auto_onoff2.pressed)
                {
                        auto_onoff2.down();
                }
                else auto_onoff2.up();
        }

        if (which=='2')
        {
                if (!manual_control2.pressed) manual_control2.down();
                else manual_control2.up();
        }

        if (which=='3')
        {
                if (!name_tags2.pressed)
                {
                name_tags2.down();
                goto Page_3;
                }
                else name_tags2.up();
        }


        if (which=='4')
        {
                if (!voice_tags2.pressed)
                {
                voice_tags2.down();
                goto Page_3;
                }
                else voice_tags2.up();
        }

        if (which=='5')
        {
```

```
                if (!custom_session2.pressed) custom_session2.down();
                else custom_session2.up();
        }

        if (which=='6')
        {
                if (!memory_framing2.pressed)
                {
                memory_framing2.down();
                goto Page_3;
                }
                else memory_framing2.up();
        }

        if (which=='7')
        {
                if (!remote_commands2.pressed) remote_commands2.down();
                else remote_commands2.up();
        }

        if (which=='8')
        {
                if (!av_options2.pressed) av_options2.down();
                else av_options2.up();
        }

        if (which=='9')
        {
                if (!codec_control2.pressed) codec_control2.down();
                else codec_control2.up();
        }

        if (which=='a')
        {
                if (!mcu_control2.pressed) mcu_control2.down();
                else mcu_control2.up();
        }

        if (which=='b')
        {
                if (!dial_connects2.pressed)
                {
                dial_connects2.down();
                goto Page_6;
                }
                else dial_connects2.up();
        }

        if (which=='c')
        {
                if (!mute2.pressed) mute2.down();
                else mute2.up();
        }

if (which=='S') Page_2_flag=FALSE;
}
goto pgm_terminate;


Page_3:
//      this is the first "individual identification" menu.
//      and includes the step into nametags.
```

```
        setfillstyle(INTERLEAVE_FILL,DARKGRAY);
        bar3d(0,0,getmaxx(),getmaxy(),0,0);
        ind_id3.up();
        video_screen.up();
        video_message1.down();
        time_date2.up();
        frame_cam3.up();
        cam_preset3.up();
        name_tags2.up();
        voice_tags2.up();
        autofollow3.up();
        return3.up();
        mute2.up();

        Page_3_flag=TRUE;

while (Page_3_flag)
{
//      temporary keypad substitute for the touch screen.
which = getch();

        if (which=='1')
        {
                if (!frame_cam3.pressed)
                {
                        frame_cam3.down();
                }
                else frame_cam3.up();
        }

        if (which=='2')
        {
                if (!cam_preset3.pressed) cam_preset3.down();
                else cam_preset3.up();
        }

        if (which=='3')
        {
                if (!name_tags2.pressed)
                {
                        name_tags2.down();
                        touch_face3.up();
                type_id3.up();
                write_id3.up();
                cancel3.up();

                type_or_write:
                which=getch();
                // the cancel button has been pressed.
                if (which=='9') goto Page_3;
                // type nametags.
                if (which=='x') goto Page_4;
                // write nametags.
                if (which=='y') goto Page_5;
                goto type_or_write;
        }

                else name_tags2.up();
}


        if (which=='4')
        {
```

```
                if (!voice_tags2.pressed) voice_tags2.down();
//              goto Page_4;
        else voice_tags2.up();
        }


        if (which==='5')
        {
                if (!autofollow3.pressed) autofollow3.down();
//              goto Page_4;
                else autofollow3.up();
        }


        if (which==='b')
        {
                if (!return3.pressed) return3.down();
                goto Page_2;
        }
                else return3.up();


        if (which==='c')
        {
                if (!mute2.pressed) mute2.down();
                else mute2.up();
        }

if (which==='S') Page_3_flag=FALSE;
}
goto pgm_terminate;


Page_4:
//      this is the nametags typing page.

        setfillstyle(INTERLEAVE_FILL,DARKGRAY);
        bar3d(0,0,getmaxx(),getmaxy(),0,0);
        ind_id3.up();
        video_screen.up();
        video_message1.down();
        frame_cam3.up();
        cam_preset3.up();
        name_tags2.down();
        voice_tags2.up();
        autofollow3.up();
        return3.up();
        mute2.up();
        keyboard.up();
    typing_done.up();

        Page_4_flag=TRUE;

while (Page_4_flag)
{
//      temporary keypad substitute for the touch screen.
which = getch();

        if (which==='7')
        {
                if (!typing_done.pressed) typing_done.down();
                goto Page_3;
        }
```

```
                else typing_done.up();


        if (which=='b')
        {
                if (!return3.pressed) return3.down();
                goto Page_3;
        }
                else return3.up();


        if (which=='c')
        {
                if (!mute2.pressed) mute2.down();
                else mute2.up();
        }

if (which=='S') Page_4_flag=FALSE;
}
goto pgm_terminate;


Page_5:
//      this is the nametags writing page.

        setfillstyle(INTERLEAVE_FILL,DARKGRAY);
        bar3d(0,0,getmaxx(),getmaxy(),0,0);
        ind_id3.up();
        video_screen.up();
        video_message1.down();
        frame_cam3.up();
        cam_preset3.up();
        name_tags2.down();
        voice_tags2.up();
        autofollow3.up();
        return3.up();
        mute2.up();
        writing_space.up();
    writing_done.up();

        Page_5_flag=TRUE;

while (Page_5_flag)
{
//      temporary keypad substitute for the touch screen.
which = getch();

        if (which=='7')
        {
                if (!typing_done.pressed) typing_done.down();
                goto Page_3;
        }
                else typing_done.up();


        if (which=='b')
        {
                if (!return3.pressed) return3.down();
                goto Page_3;
        }
                else return3.up();
```

```
            if (which == 'c')
            {
                    if (!mute2.pressed) mute2.down();
                    else mute2.up();
            }

    if (which == 'S') Page_5_flag = FALSE;
    }
    goto pgm_terminate;

    Page_6:
    //      this is the connections dialing and directory maintenance page.

            setfillstyle(INTERLEAVE_FILL,DARKGRAY);
            bar3d(0,0,getmaxx(),getmaxy(),0,0);
            dial_connects6.up();
        directory6.up();
        keypad6.up();
        scroll_up6.up();
        scroll_down6.up();
        dial_this6.up();
        add_entry6.up();
        delete_entry6.up();
            manual_dialing6.up();
        line_16.up();
        line_26.up();
            dial_tone6.up();
        hang_up6.up();
            return3.up();
            mute2.up();

            Page_6_flag = TRUE;

    while (Page_6_flag)
    {
    //      temporary keypad substitute for the touch screen.
    which = getch();

            if (which == 'h')
            {
                    if (!return3.pressed)
                    {
                    return3.down();
                    goto Page_2;
                }
            }
                    else return3.up();


            if (which == 'c')
            {
                    if (!mute2.pressed) mute2.down();
                    else mute2.up();
            }

    if (which == 'S') Page_6_flag = FALSE;
    }
    goto pgm_terminate;

    pgm_terminate:
    getch();
```

18.1.97

```
//          this is the closing sequence.

closegraph();
}
```

```
/*****************************************/
/*        ARPROCES.H                     */
/*   Image Processing Header File        */
/*    Area Processing Functions          */
/*     written in Turbo C 2.0            */
/*****************************************/

/* Area Process Function Prototypes */
CompletionCode Convolution(BYTE huge *InImage, unsigned Col, unsigned Row,
                           unsigned Width, unsigned Height,
                           short *Kernel, unsigned KernelCols,
                           unsigned KernelRows, unsigned Scale,
                           unsigned Absolute, BYTE huge * *OutImageBufPtr);

CompletionCode RealConvolution(BYTE huge *InImage,
                               unsigned Col, unsigned Row,
                               unsigned Width, unsigned Height,
                               double *Kernel, unsigned KernelCols,
                               unsigned KernelRows, unsigned Scale,
                               unsigned Absolute, BYTE huge * *OutImageBufPtr);

CompletionCode MedianFilter(BYTE huge *InImage, unsigned Col, unsigned Row,
                            unsigned Width, unsigned Height,
                            unsigned NeighborhoodCols, unsigned NeighborhoodRows,
                            BYTE huge * *OutImageBufPtr);

CompletionCode SobelEdgeDet(BYTE huge *InImage,
                            unsigned Col, unsigned Row,
                            unsigned Width, unsigned Height,
                            unsigned Threshold, unsigned Overlay,
                            BYTE huge * *OutImageBufPtr);
```

```
/*****************************************/
/*       ARPROCES.C                      */
/*    Image Processing Code              */
/*   Area Processing Functions           */
/*     written in Turbo C 2.0            */
/*****************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <process.h>
#include <math.h>
#include <graphics.h>
#include "misc.h"
#include "pcx.h"
#include "vga.h"
#include "imagesup.h"
#include "arprocess.h"


/*
Integer Convolution Function
*/

CompletionCode Convolution(BYTE huge *InImage, unsigned Col, unsigned Row,
                           unsigned Width, unsigned Height,
                           short *Kernel, unsigned KernelCols,
                           unsigned KernelRows, unsigned Scale,
                           unsigned Absolute, BYTE huge * *OutImageBufPtr)
{
  register unsigned ColExtent, RowExtent;
  register unsigned ImageCol, ImageRow, KernCol, KernRow;
  unsigned ColOffset, RowOffset, TempCol, TempRow;
  BYTE huge *OutputImageBuffer;
  long Sum;
  short *KernelPtr;

  if (ParameterCheckOK(Col,Row,Col+Width,Row+Height,"Convolution"))
  {
    /* Image must be at least the same size as the kernel */
    if (Width >= KernelCols && Height >= KernelRows)
    {
      /* allocate far memory buffer for output image */
      OutputImageBuffer = (BYTE huge *)
                          farcalloc(RASTERSIZE,(unsigned long)sizeof(BYTE));

      if (OutputImageBuffer == NULL)
      {
        restorecrtmode();
        printf("Error Not enough memory for convolution output buffer\n");
        return (ENoMemory);
      }

      /* Store address of output image buffer */
      *OutImageBufPtr = OutputImageBuffer;

      /*
      Clearing the output buffer to white will show the
      boarder areas not touched by the convolution. It also
```

```
        provides a nice white frame for the output image.
        */

        ClearImageArea(OutputImageBuffer,MINCOLNUM,MINROWNUM,
                       MAXCOLS,MAXROWS,WHITE);

        ColOffset = KernelCols/2;
        RowOffset = KernelRows/2;
        /* Compensate for edge effects */
        Col += ColOffset;
        Row += RowOffset;
        Width -= (KernelCols - 1);
        Height -= (KernelRows - 1);

        /* Calculate new range of pixels to act upon */
        ColExtent = Col + Width;
        RowExtent = Row + Height;

        for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
        {
          TempRow = ImageRow - RowOffset;
          for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
          {
            TempCol = ImageCol - ColOffset;
            Sum = 0L;
            KernelPtr = Kernel;
            for (KernCol = 0; KernCol < KernelCols; KernCol++)
                for (KernRow = 0; KernRow < KernelRows; KernRow++)
                  Sum += (GetPixelFromImage(InImage,
                          TempCol+KernCol, TempRow+KernRow) *
                          (*KernelPtr++));

            /* If absolute value is requested */
            if (Absolute)
                Sum = labs(Sum);

            /* Summation performed: Scale and range Sum*/
            Sum >>= (long) Scale;

            Sum = (Sum < MINSAMPLEVAL) ? MINSAMPLEVAL:Sum;
            Sum = (Sum > MAXSAMPLEVAL) ? MAXSAMPLEVAL:Sum;
            PutPixelInImage(OutputImageBuffer,ImageCol,ImageRow,(BYTE)Sum);
          }
        }
      }
      else
        return(EKernelSize);
    }
    return(NoError);
}

/*
Real Number Convolution Function. This convolution function is
only used when the kernel entries are floating point numbers
instead of integers. Because of the floating point operations
envolved, this function is substantially slower than the already
slow integer version above.
*/

CompletionCode RealConvolution(BYTE huge *InImage,
                               unsigned Col, unsigned Row,
                               unsigned Width, unsigned Height,
                               double *Kernel, unsigned KernelCols,
```

```
                                unsigned KernelRows, unsigned Scale,
                                unsigned Absolute, BYTE huge * *OutImageBufPtr)
{
 register unsigned ColExtent, RowExtent;
 register unsigned ImageCol, ImageRow, KernCol, KernRow;
 unsigned ColOffset, RowOffset, TempCol, TempRow;
 BYTE huge *OutputImageBuffer;
 double Sum;
 double *KernelPtr;


 if (ParameterCheckOK(Col,Row,Col+Width,Row+Height,"Convolution"))
 {
   /* Image must be at least the same size as the kernel */
   if (Width >= KernelCols && Height >= KernelRows)
   {
        /* allocate far memory buffer for output image */
        OutputImageBuffer = (BYTE huge *)
                            farcalloc(RASTERSIZE,(unsigned long)sizeof(BYTE));

        if (OutputImageBuffer == NULL)
        {
          restorecrtmode();
          printf("Error Not enough memory for convolution output buffer\n");
          return (ENoMemory);
        }

        /* Store address of output image buffer */
        *OutImageBufPtr = OutputImageBuffer;

        /*
        Clearing the output buffer to white will show the
        boarder areas not touched by the convolution. It also
        provides a nice white frame for the output image.
        */

        ClearImageArea(OutputImageBuffer,MINCOLNUM,MINROWNUM,
                       MAXCOLS,MAXROWS,WHITE);

        ColOffset = KernelCols/2;
        RowOffset = KernelRows/2;
        /* Compensate for edge effects */
        Col += ColOffset;
        Row += RowOffset;
        Width -= (KernelCols - 1);
        Height -= (KernelRows - 1);

        /* Calculate new range of pixels to act upon */
        ColExtent = Col + Width;
        RowExtent = Row + Height;

        for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
        {
          TempRow = ImageRow - RowOffset;
          for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
          {
            TempCol = ImageCol - ColOffset;
            Sum = 0.0;
            KernelPtr = Kernel;
            for (KernCol = 0; KernCol < KernelCols; KernCol++)
                 for (KernRow = 0; KernRow < KernelRows; KernRow++)
                   Sum += (GetPixelFromImage(InImage,
                           TempCol+KernCol, TempRow+KernRow) *
```

```
                        (*KernelPtr++));

                /* If absolute value is requested */
                if (Absolute)
                    Sum = fabs(Sum);

                /* Summation performed. Scale and range Sum */
                Sum /= (double)(1 << Scale);

                Sum = (Sum < MINSAMPLEVAL) ? MINSAMPLEVAL:Sum;
                Sum = (Sum > MAXSAMPLEVAL) ? MAXSAMPLEVAL:Sum;
                PutPixelInImage(OutputImageBuffer,ImageCol,ImageRow,(BYTE)Sum);
              }
            }
          }
          else
              return(EKernelSize);
        }
        return(NoError);
}


/*
Byte compare for use with the qsort library function call
in the Median filter function.
*/

int ByteCompare(BYTE *Entry1, BYTE *Entry2)
{
  if (*Entry1 < *Entry2)
    return(-1);
  else if (*Entry1 > *Entry2)
    return(1);
  else
    return(0);
}


CompletionCode MedianFilter(BYTE huge *InImage, unsigned Col, unsigned Row,
                            unsigned Width, unsigned Height,
                            unsigned NeighborhoodCols, unsigned NeighborhoodRows,
                            BYTE huge * *OutImageBufPtr)
{
  register unsigned ColExtent, RowExtent;
  register unsigned ImageCol, ImageRow, NeighborCol, NeighborRow;
  unsigned ColOffset, RowOffset, TempCol, TempRow, PixelIndex;
  unsigned TotalPixels, MedianIndex;
  BYTE huge *OutputImageBuffer;
  BYTE *PixelValues;


  if (ParameterCheckOK(Col,Row,Col+Width,Row+Height,"Median Filter"))
  {
    /* Image must be at least the same size as the neighborhood */
    if (Width >= NeighborhoodCols && Height >= NeighborhoodRows)
    {
        /* allocate far memory buffer for output image */
        OutputImageBuffer = (BYTE huge *)
                            farcalloc(RASTERSIZE,(unsigned long)sizeof(BYTE));

        if (OutputImageBuffer == NULL)
        {
```

```
        restorecrtmode();
        printf("Error Not enough memory for median filter output buffer\n");
        return (ENoMemory);
    }

    /* Store address of ouput image buffer */
    *OutImageBufPtr = OutputImageBuffer;

    /*
    Clearing the output buffer to white will show the
    boarder areas not touched by the median filter. It also
    provides a nice white frame for the output image.
    */

    ClearImageArea(OutputImageBuffer,MINCOLNUM,MINROWNUM,
                   MAXCOLS,MAXROWS,WHITE);

    /* Calculate border pixel to miss */
    ColOffset = NeighborhoodCols/2;
    RowOffset = NeighborhoodRows/2;

    /* Compensate for edge effects */
    Col += ColOffset;
    Row += RowOffset;
    Width -= (NeighborhoodCols - 1);
    Height -= (NeighborhoodRows - 1);

    /* Calculate new range of pixels to act upon */
    ColExtent = Col + Width;
    RowExtent = Row + Height;

    TotalPixels = (NeighborhoodCols*NeighborhoodRows);
    MedianIndex = (NeighborhoodCols*NeighborhoodRows)/2;

    /* allocate memory for pixel buffer */
    PixelValues = (BYTE *) calloc(TotalPixels,(unsigned)sizeof(BYTE));

    if (PixelValues == NULL)
    {
      restorecrtmode();
      printf("Error Not enough memory for median filter pixel buffer\n");
      return (ENoMemory);
    }

    for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
    {
      TempRow = ImageRow - RowOffset;
      for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
      {
        TempCol = ImageCol - ColOffset;
        PixelIndex = 0;
        for (NeighborCol = 0; NeighborCol < NeighborhoodCols; NeighborCol++)
            for (NeighborRow = 0; NeighborRow < NeighborhoodRows; NeighborRow++)
              PixelValues[PixelIndex++] =
                GetPixelFromImage(InImage,TempCol+NeighborCol,
                                  TempRow+NeighborRow);

        /*
        Quick sort the brightness values into ascending order
        and then pick out the median or middle value as
        that for the pixel.
        */
        qsort(PixelValues,TotalPixels,sizeof(BYTE),ByteCompare);
```

```
                PutPixelInImage(OutputImageBuffer,ImageCol,ImageRow,
                                PixelValues[MedianIndex]);
            }
        }
    }
    else
        return(EKernelSize);
  }
  free(PixelValues);      /* give up the pixel value buffer */
  return(NoError);
}

/*
Sobel Edge Detection Function
*/

CompletionCode SobelEdgeDet(BYTE huge *InImage,
                            unsigned Col, unsigned Row,
                            unsigned Width, unsigned Height,
                            unsigned Threshold, unsigned Overlay,
                            BYTE huge * *OutImageBufPtr)
{
  register unsigned ColExtent, RowExtent;
  register unsigned ImageCol, ImageRow;
  unsigned PtA, PtB, PtC, PtD, PtE, PtF, PtG, PtH, PtI;
  unsigned LineAEIAveAbove, LineAEIAveBelow, LineAEIMaxDif;
  unsigned LineBEHAveAbove, LineBEHAveBelow, LineBEHMaxDif;
  unsigned LineCEGAveAbove, LineCEGAveBelow, LineCEGMaxDif;
  unsigned LineDEFAveAbove, LineDEFAveBelow, LineDEFMaxDif;
  unsigned MaxDif;
  BYTE huge *OutputImageBuffer;


  if (ParameterCheckOK(Col,Row,Col+Width,Row+Height,"Sobel Edge Detector"))
  {
    /* allocate far memory buffer for output image */
    OutputImageBuffer = (BYTE huge *)
                        farcalloc(RASTERSIZE,(unsigned long)sizeof(BYTE));

    if (OutputImageBuffer == NULL)
    {
        restorecrtmode();
        printf("Error Not enough memory for Sobel output buffer\n");
        return (ENoMemory);
    }

    /* Store address of output image buffer */
    *OutImageBufPtr = OutputImageBuffer;

    /*
    Clearing the output buffer
    */

    ClearImageArea(OutputImageBuffer,MINCOLNUM,MINROWNUM,
                   MAXCOLS,MAXROWS,BLACK);

    /* Compensate for edge effects of 3x3 pixel neighborhood */
    Col += 1;
    Row += 1;
    Width -= 2;
    Height -= 2;

    /* Calculate new range of pixels to act upon */
```

```
    ColExtent = Col + Width;
    RowExtent = Row + Height;

    for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
        for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
        {
            /* Get each pixel in 3x3 neighborhood */
            PtA = GetPixelFromImage(InImage,ImageCol-1,ImageRow-1);
            PtB = GetPixelFromImage(InImage,ImageCol  ,ImageRow-1);
            PtC = GetPixelFromImage(InImage,ImageCol+1,ImageRow-1);
            PtD = GetPixelFromImage(InImage,ImageCol-1,ImageRow  );
            PtE = GetPixelFromImage(InImage,ImageCol  ,ImageRow  );
            PtF = GetPixelFromImage(InImage,ImageCol+1,ImageRow  );
            PtG = GetPixelFromImage(InImage,ImageCol-1,ImageRow+1);
            PtH = GetPixelFromImage(InImage,ImageCol  ,ImageRow+1);
            PtI = GetPixelFromImage(InImage,ImageCol+1,ImageRow+1);

            /*
            Calculate average above and below the line.
            Take the absolute value of the difference.
            */
            LineAEIAveBelow = (PtD+PtG+PtH)/3;
            LineAEIAveAbove = (PtB+PtC+PtF)/3;
            LineAEIMaxDif = abs(LineAEIAveBelow-LineAEIAveAbove);

            LineBEHAveBelow = (PtA+PtD+PtG)/3;
            LineBEHAveAbove = (PtC+PtF+PtI)/3;
            LineBEHMaxDif = abs(LineBEHAveBelow-LineBEHAveAbove);

            LineCEGAveBelow = (PtF+PtH+PtI)/3;
            LineCEGAveAbove = (PtA+PtB+PtD)/3;
            LineCEGMaxDif = abs(LineCEGAveBelow-LineCEGAveAbove);

            LineDEFAveBelow = (PtG+PtH+PtI)/3;
            LineDEFAveAbove = (PtA+PtB+PtC)/3;
            LineDEFMaxDif = abs(LineDEFAveBelow-LineDEFAveAbove);
            /*
            Find the maximum value of the absolute differences
            from the four possibilities.
            */
            MaxDif = MAX(LineAEIMaxDif,LineBEHMaxDif);
            MaxDif = MAX(LineCEGMaxDif,MaxDif);
            MaxDif = MAX(LineDEFMaxDif,MaxDif);
            /*
            If maximum difference is above the threshold, set
            the pixel of interest (center pixel) to white. If
            below the threshold optionally copy the input image
            to the output image. This copying is controlled by
            the parameter Overlay.
            */
            if (MaxDif >= Threshold)
              PutPixelInImage(OutputImageBuffer,ImageCol,ImageRow,WHITE);
            else if (Overlay)
              PutPixelInImage(OutputImageBuffer,ImageCol,ImageRow,PtE);
        }
    }
    return(NoError);
}
```

18.1.97

```
/*****************************************/
/*          FRPOCES.H                    */
/*    Image Processing Header File       */
/*    Frame Processing Functions         */
/*      written in Turbo C 2.0           */
/*****************************************/

/* User defined image combination type */
typedef enum {And,Or,Xor,Add,Sub,Mult,Div,Min,Max,Ave,Overlay} BitFunction;

/* Frame Process Function Prototypes */

void CombineImages(BYTE huge *SImage,
                   unsigned SCol, unsigned SRow,
                   unsigned SWidth, unsigned SHeight,
                   BYTE huge *DImage,
                   unsigned DCol, unsigned DRow,
                   enum BitFunction CombineType,
                   short Scale);
```

```
/*************************************/
/*          FPROCES.C                */
/*      Image Processing Code        */
/*     Frame Process Functions       */
/*      written in Turbo C 2.0       */
/*************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include "misc.h"
#include "pcx.h"
#include "vga.h"
#include "imagesup.h"
#include "frprocess.h"


/* Single function performs all image combinations */

void CombineImages(BYTE huge *SImage,
                   unsigned SCol, unsigned SRow,
                   unsigned SWidth, unsigned SHeight,
                   BYTE huge *DImage,
                   unsigned DCol, unsigned DRow,
                   enum BitFunction CombineType,
                   short Scale)
{
  register unsigned SImageCol, SImageRow, DestCol;
  short SData, DData;
  unsigned SColExtent, SRowExtent;

  if (ParameterCheckOK(SCol,SRow,SCol+SWidth,SRow+SHeight,"CombineImages") &&
     ParameterCheckOK(DCol,DRow,DCol+SWidth,DRow+SHeight,"CombineImages"))
  {
    SColExtent = SCol+SWidth;
    SRowExtent = SRow+SHeight;

    for (SImageRow = SRow; SImageRow < SRowExtent; SImageRow++)
    {
        /* Reset the destination Column count every row */
        DestCol = DCol;
        for (SImageCol = SCol; SImageCol < SColExtent; SImageCol++)
        {
          /* Get a byte of the source and dest image data */
          SData = GetPixelFromImage(SImage,SImageCol,SImageRow);
          DData = GetPixelFromImage(DImage,DestCol,DRow);

          /* Combine source and dest data according to parameter */
          switch(CombineType)
          {
            case And:
                DData &= SData;
                break;
            case Or:
                DData |= SData;
                break;
            case Xor:
                DData ^= SData;
                break;
```

```
            case Add:
                DData += SData;
                break;
            case Sub:
                DData -= SData;
                break;
            case Mult:
                DData *= SData;
                break;
            case Div:
                if (SData != 0)
                  DData /= SData;
                break;
            case Min:
                DData = MIN(SData,DData);
                break;
            case Max:
                DData = MAX(SData,DData);
                break;
            case Ave:
                DData = (SData+DData)/2;
                break;
            case Overlay:
                DData = SData;
                break;
          }
          /*
          Scale the resultant data if requested to. A positive
          Scale value shifts the destination data to the right
          thereby dividing it by a power of two. A zero Scale
          value leaves the data untouched. A negative Scale
          value shifts the data left thereby multiplying it by
          a power of two.
          */
          if (Scale < 0)
            DData << = abs(Scale);
          else if (Scale > 0)
            DData > > = Scale;
          /* Don't let the pixel data get out of range */
          DData = (DData < MINSAMPLEVAL) ? MINSAMPLEVAL:DData;
          DData = (DData > MAXSAMPLEVAL) ? MAXSAMPLEVAL:DData;
          PutPixelInImage(DImage,DestCol++,DRow,DData);
        }
        /* Bump to next row in the destination image */
        DRow++;
      }
    }
  }
```

```
/*****************************************/
/*          GEPROCES.H                   */
/*    Image Processing Header File       */
/*   Geometric Processing Functions      */
/*      written in Turbo C 2.0           */
/*****************************************/


/* Misc user defined types */
typedef enum {HorizMirror, VertMirror} MirrorType;


/* Geometric processes function prototypes */
void ScaleImage(BYTE huge *InImage, unsigned SCol, unsigned SRow,
                unsigned SWidth, unsigned SHeight,
                double ScaleH, double ScaleV,
                BYTE huge *OutImage,
                unsigned DCol, unsigned DRow,
                unsigned Interpolate);

void SizeImage(BYTE huge *InImage, unsigned SCol, unsigned SRow,
               unsigned SWidth, unsigned SHeight,
               BYTE huge *OutImage,
               unsigned DCol, unsigned DRow,
               unsigned DWidth, unsigned DHeight,
               unsigned Interpolate);

void RotateImage(BYTE huge *InImage, unsigned Col, unsigned Row,
                 unsigned Width, unsigned Height, double Angle,
                 BYTE huge *OutImage, unsigned Interpolate);

void TranslateImage(BYTE huge *InImage,
                    unsigned SCol, unsigned SRow,
                    unsigned SWidth, unsigned SHeight,
                    BYTE huge *OutImage,
                    unsigned DCol, unsigned DRow,
                    unsigned EraseFlag);

void MirrorImage(BYTE huge *InImage,
                 unsigned SCol, unsigned SRow,
                 unsigned SWidth, unsigned SHeight,
                 enum MirrorType WhichMirror,
                 BYTE huge *OutImage,
                 unsigned DCol, unsigned DRow);
```

```
/*****************************************/
/*          GEPROCES.C                   */
/*      Image Processing Code            */
/*  Geometric Processing Functions       */
/*      written in Turbo C 2.0           */
/*****************************************/

#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <process.h>
#include <math.h>
#include <graphics.h>
#include "misc.h"
#include "pcx.h"
#include "vga.h"
#include "imagesup.h"


void ScaleImage(BYTE huge *InImage, unsigned SCol, unsigned SRow,
                unsigned SWidth, unsigned SHeight,
                double ScaleH, double ScaleV,
                BYTE huge *OutImage,
                unsigned DCol, unsigned DRow,
                unsigned Interpolate)
{
  unsigned DestWidth, DestHeight;
  unsigned PtA, PtB, PtC, PtD, PixelValue;
  register unsigned SPixelColNum, SPixelRowNum, DestCol, DestRow;
  double SPixelColAddr, SPixelRowAddr;
  double ColDelta, RowDelta;
  double ContribFromAandB, ContribFromCandD;

  DestWidth =  ScaleH * SWidth + 0.5;
  DestHeight = ScaleV * SHeight + 0.5;

  if (ParameterCheckOK(SCol,SRow,SCol+SWidth,SRow+SHeight,"ScaleImage") &&
     ParameterCheckOK(DCol,DRow,DCol+DestWidth,DRow+DestHeight,"ScaleImage"))
  {
    /* Calculations from destination perspective */
    for (DestRow = 0; DestRow < DestHeight; DestRow++)
    {
        SPixelRowAddr = DestRow/ScaleV;
        SPixelRowNum  = (unsigned) SPixelRowAddr;
        RowDelta      = SPixelRowAddr - SPixelRowNum;
        SPixelRowNum += SRow;

        for (DestCol = 0; DestCol < DestWidth; DestCol++)
        {
          SPixelColAddr = DestCol/ScaleH;
          SPixelColNum  = (unsigned) SPixelColAddr;
          ColDelta      = SPixelColAddr - SPixelColNum;
          SPixelColNum += SCol;

          if (Interpolate)
          {
            /*
            SPixelColNum and SPixelRowNum now contain the pixel
            coordinates of the upper left pixel of the targetted
            pixel's (point X) neighborhood. This is point A below:
                        A     B
                          X
```

```
                        C     D
                    We must retrieve the brightness level of each of the
                    four pixels to calculate the value of the pixel put into
                    the destination image.

                    Get point A brightness as it will always lie within the
                    input image area. Check to make sure the other points are
                    within also. If so use their values for the calculations.
                    If not, set them all equal to point A's value. This induces
                    an error but only at the edges on an image.
                    */

                    PtA = GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum);
                    if (((SPixelColNum+1) < MAXCOLS) && ((SPixelRowNum+1) < MAXROWS))
                    {
                        PtB = GetPixelFromImage(InImage,SPixelColNum+1,SPixelRowNum);
                        PtC = GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum+1);
                        PtD = GetPixelFromImage(InImage,SPixelColNum+1,SPixelRowNum+1);
                    }
                    else
                    {
                        /* All points have equal brightness */
                        PtB=PtC=PtD=PtA;
                    }
                    /*
                    Interpolate to find brightness contribution of each pixel
                    in neighborhood. Done in both the horizontal and vertical
                    directions.
                    */
                    ContribFromAandB = ColDelta*((double)PtB - PtA) + PtA;
                    ContribFromCandD = ColDelta*((double)PtD - PtC) + PtC;
                    PixelValue = 0.5 + ContribFromAandB +
                                 (ContribFromCandD - ContribFromAandB)*RowDelta;
                }
                else
                  PixelValue=GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum);

                /* Put the pixel into the destination buffer */
                PutPixelInImage(OutImage,DestCol+DCol,DestRow+DRow,PixelValue);
            }
        }
    }
}


void SizeImage(BYTE huge *InImage, unsigned SCol, unsigned SRow,
               unsigned SWidth, unsigned SHeight,
               BYTE huge *OutImage,
               unsigned DCol, unsigned DRow,
               unsigned DWidth, unsigned DHeight,
               unsigned Interpolate)
{
  double HScale, VScale;

  /* Check for parameters out of range */
  if (ParameterCheckOK(SCol,SRow,SCol+SWidth,SRow+SHeight,"SizeImage") &&
      ParameterCheckOK(DCol,DRow,DCol+DWidth,DRow+DHeight,"SizeImage"))
  {
    /*
    Calculate horizontal and vertical scale factors required
    to fit specified portion of input image into specified portion
    of output image.
    */
```

```
    HScale = (double)DWidth/(double)SWidth;
    VScale = (double)DHeight/(double)SHeight;

    /* Call ScaleImage to do the actual work */
    ScaleImage(InImage,SCol,SRow,SWidth,SHeight,HScale,VScale,
                OutImage,DCol,DRow,Interpolate);
  }
}

void RotateImage(BYTE huge *InImage, unsigned Col, unsigned Row,
                 unsigned Width, unsigned Height, double Angle,
                 BYTE huge *OutImage, unsigned Interpolate)
{
  register unsigned ImageCol, ImageRow;
  unsigned CenterCol, CenterRow, SPixelColNum, SPixelRowNum;
  unsigned ColExtent, RowExtent, PixelValue;
  unsigned PtA, PtB, PtC, PtD;
  double   DPixelRelativeColNum, DPixelRelativeRowNum;
  double   CosAngle, SinAngle, SPixelColAddr, SPixelRowAddr;
  double   ColDelta, RowDelta;
  double   ContribFromAandB, ContribFromCandD;

  if (ParameterCheckOK(Col,Row,Col+Width,Row+Height,"RotateImage"))
  {
    /* Angle must be in 0..359.9 */
    while (Angle >= 360.0)
          Angle -= 360.0;

    /* Convert angle from degrees to radians */
    Angle *= ((double) 3.14159/(double) 180.0);

    /* Calculate angle values for rotation */
    CosAngle = cos(Angle);
    SinAngle = sin(Angle);

    /* Center of rotation */
    CenterCol = Col + Width/2;
    CenterRow = Row + Height/2;

    ColExtent = Col + Width;
    RowExtent = Row + Height;

    /*
    All calculations are performed from the destination image
    perspective. Absolute pixel values must be converted into
    inches of display distance to keep the aspect value
    correct when image is rotated. After rotation, the calculated
    display distance is converted back to real pixel values.
    */

    for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
    {
        DPixelRelativeRowNum = (double)ImageRow - CenterRow;
        /* Convert row value to display distance from image center */
        DPixelRelativeRowNum *= LRINCHESPERPIXELVERT;

        for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
        {
          DPixelRelativeColNum = (double)ImageCol - CenterCol;
          /* Convert col value to display distance from image center */
          DPixelRelativeColNum *= LRINCHESPERPIXELHORIZ;
          /*
          Calculate source pixel address from destination
```

```
pixels position.
*/
SPixelColAddr = DPixelRelativeColNum*CosAngle-
                DPixelRelativeRowNum*SinAngle;
SPixelRowAddr = DPixelRelativeColNum*SinAngle+
                DPixelRelativeRowNum*CosAngle;

/*
Convert from coordinates relative to image
center back into absolute coordinates.
*/
/* Convert display distance to pixel location */
SPixelColAddr *= LRPIXELSPERINCHHORIZ;
SPixelColAddr += CenterCol;
SPixelRowAddr *= LRPIXELSPERINCHVERT;
SPixelRowAddr += CenterRow;

SPixelColNum = (unsigned) SPixelColAddr;
SPixelRowNum = (unsigned) SPixelRowAddr;
ColDelta = SPixelColAddr - SPixelColNum;
RowDelta = SPixelRowAddr - SPixelRowNum;

if (Interpolate)
{
  /*
  SPixelColNum and SPixelRowNum now contain the pixel
  coordinates of the upper left pixel of the targeted
  pixel's (point X) neighborhood. This is point A below:
                  A     B
                    X
                  C     D
  We must retrieve the brightness level of each of the
  four pixels to calculate the value of the pixel put into
  the destination image.

  Get point A brightness as it will always lie within the
  input image area. Check to make sure the other points are
  within also. If so use their values for the calculations.
  If not, set them all equal to point A's value. This induces
  an error but only at the edges on an image.
  */

  PtA = GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum);
  if (((SPixelColNum+1) < MAXCOLS) && ((SPixelRowNum+1) < MAXROWS))
  {
       PtB = GetPixelFromImage(InImage,SPixelColNum+1,SPixelRowNum);
       PtC = GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum+1);
       PtD = GetPixelFromImage(InImage,SPixelColNum+1,SPixelRowNum+1);
  }
  else
  {
       /* All points have equal brightness */
       PtB=PtC=PtD=PtA;
  }
  /*
  Interpolate to find brightness contribution of each pixel
  in neighborhood. Done in both the horizontal and vertical
  directions.
  */
  ContribFromAandB = ColDelta*((double)PtB - PtA) + PtA;
  ContribFromCandD = ColDelta*((double)PtD - PtC) + PtC;
  PixelValue = 0.5 + ContribFromAandB +
               (ContribFromCandD - ContribFromAandB)*RowDelta;
```

```
            }
            else
              PixelValue=GetPixelFromImage(InImage,SPixelColNum,SPixelRowNum);

            /* Put the pixel into the destination buffer */
            PutPixelInImage(OutImage,ImageCol,ImageRow,PixelValue);
          }
        }
      }
    }

    /*
    Caution: images must not overlap
    */
    void TranslateImage(BYTE huge *InImage,
                        unsigned SCol, unsigned SRow,
                        unsigned SWidth, unsigned SHeight,
                        BYTE huge *OutImage,
                        unsigned DCol, unsigned DRow,
                        unsigned EraseFlag)
    {
      register unsigned SImageCol, SImageRow, DestCol;
      unsigned SColExtent, SRowExtent;

      /* Check for parameters out of range */
      if (ParameterCheckOK(SCol,SRow,SCol+SWidth,SRow+SHeight,"TranslateImage") &&
          ParameterCheckOK(DCol,DRow,DCol+SWidth,DRow+SHeight,"TranslateImage"))
      {
        SColExtent = SCol+SWidth;
        SRowExtent = SRow+SHeight;

        for (SImageRow = SRow; SImageRow < SRowExtent; SImageRow++)
        {
            /* Reset the destination Column count every row */
            DestCol = DCol;
            for (SImageCol = SCol; SImageCol < SColExtent; SImageCol++)
            {
              /* Transfer byte of the image data between buffers */
              PutPixelInImage(OutImage,DestCol++,DRow,
                              GetPixelFromImage(InImage,SImageCol,SImageRow));
            }
            /* Bump to next row in the destination image */
            DRow++;
        }
        /* If erasure specified, blot out original image */
        if (EraseFlag)
            ClearImageArea(InImage,SCol,SRow,SWidth,SHeight,BLACK);
      }
    }

    void MirrorImage(BYTE huge *InImage,
                     unsigned SCol, unsigned SRow,
                     unsigned SWidth, unsigned SHeight,
                     enum MirrorType WhichMirror,
                     BYTE huge *OutImage,
                     unsigned DCol, unsigned DRow)
    {
      register unsigned SImageCol, SImageRow, DestCol;
      unsigned SColExtent, SRowExtent;

      /* Check for parameters out of range */
      if (ParameterCheckOK(SCol,SRow,SCol+SWidth,SRow+SHeight,"MirrorImage") &&
```

```
ParameterCheckOK(DCol,DRow,DCol+SWidth,DRow+SHeight,"MirrorImage"))
{
SColExtent = SCol+SWidth;
SRowExtent = SRow+SHeight;

switch(WhichMirror)
{
    case HorizMirror:
      for (SImageRow = SRow; SImageRow < SRowExtent; SImageRow++)
      {
        /* Reset the destination Column count every row */
        DestCol = DCol + SWidth;
        for (SImageCol = SCol; SImageCol < SColExtent; SImageCol++)
        {
            /* Transfer byte of the image data between buffers */
            PutPixelInImage(OutImage,--DestCol,DRow,
                    GetPixelFromImage(InImage,SImageCol,SImageRow));
        }
        /* Bump to next row in the destination image */
        DRow++;
      }
      break;
   case VertMirror:
      DRow += (SHeight-1);
      for (SImageRow = SRow; SImageRow < SRowExtent; SImageRow++)
      {
        /* Reset the destination Column count every row */
        DestCol = DCol;
        for (SImageCol = SCol; SImageCol < SColExtent; SImageCol++)
        {
            /* Transfer byte of the image data between buffers */
            PutPixelInImage(OutImage,DestCol++,DRow,
                    GetPixelFromImage(InImage,SImageCol,SImageRow));
        }
        /* Bump to next row in the destination image */
        DRow--;
      }
      break;
  }
}
}
```

```
/*****************************************/
/*          IMAGESUP.H                   */
/*   Image Processing Header File        */
/* Image Processing Support Functions    */
/*      written in Turbo C 2.0           */
/*****************************************/

/*
This file includes the general equates used for all of the
image processing code in part two of this book. Throughout
these equates, a 320x200 256 color image is assumed. If the
resolution of the processed pictures change, the equates
MAXCOLS and MAXROWS must change accordingly.
*/

/* Pixel Sample Information and Equates */
#define MAXSAMPLEBITS  6        /* 6 bits from digitizer */
#define MINSAMPLEVAL   0        /* Min sample value = 0 */

/* Max num of sample values */
#define MAXQUANTLEVELS (1 << MAXSAMPLEBITS)
/* Max sample value = 63 */
#define MAXSAMPLEVAL   (MAXQUANTLEVELS-1)

/* Image Resolution Equates */
#define MINCOLNUM      0        /* Column 0 */
#define MAXCOLS        LRMAXCOLS  /* 320 total columns */
#define MAXCOLNUM      (MAXCOLS-1) /* Last column is 319 */
#define MINROWNUM      0        /* Row 0 */
#define MAXROWS        LRMAXROWS  /* 200 total rows */
#define MAXROWNUM      (MAXROWS-1) /* Last row is 199 */

#define RASTERSIZE ((long)MAXCOLS * MAXROWS)
#define MAXNUMGRAYCOLORS MAXQUANTLEVELS

/* histogram equates */
#define HISTOCOL       0
#define HISTOROW       0
#define HISTOWIDTH     134
#define HISTOHEIGHT    84

#define BLACK          0
#define WHITE          63

#define AXISCOL        (HISTOCOL+3)
#define AXISROW        (HISTOROW+HISTOHEIGHT-5)
#define AXISLENGTH     MAXQUANTLEVELS*2-1
#define DATACOL        AXISCOL
#define DATAROW        AXISROW-1
#define MAXDEFLECTION (HISTOHEIGHT-10)

/* External Function Declarations and Prototypes */

void CopyImage(BYTE huge *SourceBuf, BYTE huge *DestBuf);

BYTE GetPixelFromImage(BYTE huge *Image, unsigned Col, unsigned Row);

CompletionCode PutPixelInImage(BYTE huge *Image, unsigned Col,
                               unsigned Row, unsigned Color);

CompletionCode DrawHLine(BYTE huge *Image, unsigned Col, unsigned Row,
                         unsigned Length, unsigned Color);
```

```
CompletionCode DrawVLine(BYTE huge *Image, unsigned Col, unsigned Row,
                         unsigned Length, unsigned Color);

void ReadImageAreaToBuf (BYTE huge *Image, unsigned Col, unsigned Row,
                         unsigned Width, unsigned Height,
                         BYTE huge *Buffer);

void WriteImageAreaFromBuf (BYTE huge *Buffer, unsigned BufWidth,
                            unsigned BufHeight, BYTE huge *Image,
                            unsigned ImageCol, unsigned ImageRow);

void ClearImageArea(BYTE huge *Image,unsigned Col, unsigned Row,
                    unsigned Width, unsigned Height,
                    unsigned PixelValue);

CompletionCode ParameterCheckOK(unsigned Col, unsigned Row,
                    unsigned ColExtent, unsigned RowExtent,
                    char *ErrorStr);
```

```
/*************************************/
/*        IMAGESUP.C                 */
/* Image Processing Support Functions */
/*     written in Turbo C 2.0         */
/*************************************/

#include <stdio.h>
#include <process.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <mem.h>
#include <graphics.h>
#include "misc.h"
#include "pcx.h"
#include "vga.h"
#include "imagesup.h"


extern struct PCX_File PCXData;
extern unsigned ImageWidth;
extern unsigned ImageHeight;

/*
Image Processing Support Functions - See text for details.
*/

/*
Copy a complete image from source buffer to destination buffer
*/

void CopyImage(BYTE huge *SourceBuf, BYTE huge *DestBuf)
{
  movedata(FP_SEG(SourceBuf),FP_OFF(SourceBuf),
           FP_SEG(DestBuf),FP_OFF(DestBuf),
           (unsigned) RASTERSIZE);
}


/*
NOTE: to index into the image memory like an array, the index
value must be a long variable type, NOT just cast to long.
*/

BYTE GetPixelFromImage(BYTE huge *Image, unsigned Col, unsigned Row)
{
 unsigned long PixelBufOffset;

 if((Col < ImageWidth) && (Row < ImageHeight))
 {
   PixelBufOffset = Row;          /* done to prevent overflow */
   PixelBufOffset *= ImageWidth;
   PixelBufOffset += Col;
   return(Image[PixelBufOffset]);
 }
 printf("GetPixelFromImage Error: Coordinate out of range\n");
 printf(" Col = %d  Row = %d\n",Col,Row);
 return(FALSE);
}

CompletionCode PutPixelInImage(BYTE huge *Image, unsigned Col,
                               unsigned Row, unsigned Color)
{
```

```
    unsigned long PixelBufOffset;

    if((Col < ImageWidth) && (Row < ImageHeight))
    {
      PixelBufOffset  = Row;          /* done to prevent overflow */
      PixelBufOffset *= ImageWidth;
      PixelBufOffset += Col;
      Image[PixelBufOffset] = Color;
      return(TRUE);
    }
    else
    {
      printf("PutPixelInImage Error: Coordinate out of range\n");
      printf("  Col = %d  Row = %d\n",Col,Row);
      return(FALSE);
    }
  }

  /*
  NOTE: A length of 0 is one pixel on. A length of 1 is two pixels
  on. That is why length is incremented before being used.
  */

  CompletionCode DrawHLine(BYTE huge *Image, unsigned Col, unsigned Row,
                           unsigned Length, unsigned Color)
  {
    if ((Col < ImageWidth) && ((Col+Length) <= ImageWidth) &&
      (Row < ImageHeight))
    {
      Length++;
      while(Length--)
          PutPixelInImage(Image,Col++,Row,Color);
      return(TRUE);
    }
    else
    {
      printf("DrawHLine Error: Coordinate out of range\n");
      printf("  Col = %d  Row = %d  Length = %d\n",Col,Row,Length);
      return(FALSE);
    }
  }


  CompletionCode DrawVLine(BYTE huge *Image, unsigned Col, unsigned Row,
                           unsigned Length, unsigned Color)
  {
    if ((Row < ImageHeight) && ((Row+Length) <= ImageHeight) &&
      (Col < ImageWidth))
    {
      Length++;
      while(Length--)
          PutPixelInImage(Image,Col,Row++,Color);
      return(TRUE);
    }
    else
    {
      printf("DrawVLine Error: Coordinate out of range\n");
      printf("  Col = %d  Row = %d  Length = %d\n",Col,Row,Length);
      return(FALSE);
    }
  }

  void ReadImageAreaToBuf (BYTE huge *Image, unsigned Col, unsigned Row,
```

```
                                unsigned Width, unsigned Height, BYTE huge *Buffer)
{
  unsigned long PixelBufOffset = 0L;
  register unsigned ImageCol, ImageRow;

  for (ImageRow=Row; ImageRow < Row+Height; ImageRow++)
    for (ImageCol=Col; ImageCol < Col+Width; ImageCol++)
      Buffer[PixelBufOffset++] =
            GetPixelFromImage(Image,ImageCol,ImageRow);
}

void WriteImageAreaFromBuf (BYTE huge *Buffer, unsigned BufWidth,
                            unsigned BufHeight, BYTE huge *Image,
                            unsigned ImageCol, unsigned ImageRow)
{
  unsigned long PixelBufOffset;
  register unsigned BufCol, BufRow, CurrentImageCol;

  for (BufRow = 0; BufRow < BufHeight; BufRow++)
  {
    CurrentImageCol = ImageCol;
    for (BufCol = 0; BufCol < BufWidth; BufCol++)
    {
        PixelBufOffset = (unsigned long)BufRow*BufWidth+BufCol;
        PutPixelInImage(Image,CurrentImageCol,ImageRow,Buffer[PixelBufOffset]);
        CurrentImageCol++;
    }
    ImageRow++;
  }
}


void ClearImageArea(BYTE huge *Image,unsigned Col, unsigned Row,
                    unsigned Width, unsigned Height,
                    unsigned PixelValue)
{
  register unsigned BufCol, BufRow;

  for (BufRow = 0; BufRow < Height; BufRow++)
    for (BufCol = 0; BufCol < Width; BufCol++)
        PutPixelInImage(Image,BufCol+Col,BufRow+Row,PixelValue);
}


/*
This function checks to make sure the parameters passed to
the image processing functions are all within range. If so
a TRUE is returned. If not, an error message is output and
the calling program is terminated.
*/

CompletionCode ParameterCheckOK(unsigned Col, unsigned Row,
                                unsigned ColExtent, unsigned RowExtent,
                                char *FunctionName)
{
  if ((Col > MAXCOLNUM) || (Row > MAXROWNUM) ||
     (ColExtent > MAXCOLS) || (RowExtent > MAXROWS))
  {
    restorecrtmode();
    printf("Parameter(s) out of range in function: %s\n",FunctionName);
    printf("  Col = %d  Row = %d  ColExtent = %d  RowExtent = %d\n",
                Col, Row, ColExtent, RowExtent);
```

```
    exit(EBadParms);
  }
  return(TRUE);
}
```

```
/*****************************************/
/*        PTPROCES.H                     */
/*  Image Processing Header File         */
/*   Point Processing Functions          */
/*     written in Turbo C 2.0            */
/*****************************************/

extern unsigned Histogram[MAXQUANTLEVELS];

/* Function Prototypes for support and histogram functions */
void InitializeLUT(BYTE *LookUpTable);

void PtTransform(BYTE huge *ImageData, unsigned Col,
                 unsigned Row, unsigned Width,
                 unsigned Height, BYTE *LookUpTable);

void GenHistogram(BYTE huge *ImageData, unsigned Col,
                  unsigned Row, unsigned Width,
                  unsigned Height);

void DisplayHist(BYTE huge *ImageData,unsigned Col,
                 unsigned Row, unsigned Width,
                 unsigned Height);

/* Point transform functions */
void AdjImageBrightness(BYTE huge *ImageData, short BrightnessFactor,
                        unsigned Col, unsigned Row,
                        unsigned Width, unsigned Height);

void NegateImage(BYTE huge *ImageData, unsigned Threshold,
                 unsigned Col, unsigned Row,
                 unsigned Width, unsigned Height);

void ThresholdImage(BYTE huge *ImageData, unsigned Threshold,
                    unsigned Col, unsigned Row,
                    unsigned Width, unsigned Height);

void StretchImageContrast(BYTE huge *ImageData, unsigned *HistoData,
                          unsigned Threshold,
                          unsigned Col, unsigned Row,
                          unsigned Width, unsigned Height);
```

```
/*************************************/
/*        PTPROCES.C                 */
/*    Image Processing Code          */
/*   Point Process Functions         */
/*     written in Turbo C 2.0        */
/*************************************/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <dos.h>
#include <alloc.h>
#include <process.h>
#include <graphics.h>
#include "misc.h"
#include "pcx.h"
#include "vga.h"
#include "imagesup.h"


/* Histogram storage location */
unsigned Histogram[MAXQUANTLEVELS];



/*
Look Up Table (LUT) Functions

Initialize the Look Up Table (LUT) for straight through
mapping. If a point transform is performed on an initialized
LUT, output data will equal input data. This function is
usually called in preparation for modification to a LUT.
*/



void InitializeLUT(BYTE *LookUpTable)
{
  register unsigned Index;

  for (Index = 0; Index < MAXQUANTLEVELS; Index++)
    LookUpTable[Index] = Index;
}



/*
This function performs a point transform on the portion of the
image specified by Col, Row, Width and Height. The actual
transform is contained in the Look Up Table who address
is passed as a parameter.
*/



void PtTransform(BYTE huge *ImageData, unsigned Col, unsigned Row,
                 unsigned Width, unsigned Height, BYTE *LookUpTable)
{

  register unsigned ImageCol, ImageRow;
  register unsigned ColExtent, RowExtent;

  ColExtent = Col+Width;
```

```
    RowExtent = Row+Height;

    if (ParameterCheckOK(Col,Row,ColExtent,RowExtent,"PtTransform"))
      for (ImageRow=Row; ImageRow < RowExtent; ImageRow++)
          for (ImageCol=Col; ImageCol < ColExtent; ImageCol++)
            PutPixelInImage(ImageData,ImageCol,ImageRow,
                  LookUpTable[GetPixelFromImage(ImageData,ImageCol,ImageRow)]);
  }

  /* start of histogram functions

  This function calculates the histogram of any portion of an image.
  */

  void GenHistogram(BYTE huge *ImageData, unsigned Col, unsigned Row,
                    unsigned Width, unsigned Height)
  {
    register unsigned ImageRow, ImageCol, RowExtent, ColExtent;
    register unsigned Index;

    /* clear the histogram array */
    for (Index=0; Index < MAXQUANTLEVELS; Index++)
      Histogram[Index] = 0;

    RowExtent = Row+Height;
    ColExtent = Col+Width;

    if (ParameterCheckOK(Col,Row,ColExtent,RowExtent,"GenHistogram"))
    {
      /* calculate the histogram */
      for (ImageRow = Row; ImageRow < RowExtent; ImageRow++)
          for (ImageCol = Col; ImageCol < ColExtent; ImageCol++)
            Histogram[GetPixelFromImage(ImageData,ImageCol,ImageRow)] += 1;
    }
  }


  /*
  This function calculates and displays the histogram of an image
  or partial image. When called it assumes the VGA is already
  in mode 13 hex.
  */


  void DisplayHist(BYTE huge *ImageData, unsigned Col, unsigned Row,
                   unsigned Width, unsigned Height)
  {
    BYTE huge *Buffer;
    register unsigned Index, LineLength, XPos, YPos;
    unsigned MaxRepeat;

    /* Allocate enough memory to save image under histogram */
    Buffer = (BYTE huge *) farcalloc((long)HISTOWIDTH*HISTOHEIGHT,sizeof(BYTE));
    if (Buffer == NULL)
    {
      printf("No buffer memory\n");
      exit(ENoMemory);
    }
    /* Save a copy of the image */
    ReadImageAreaToBuf(ImageData,HISTOCOL,HISTOROW,HISTOWIDTH,HISTOHEIGHT,
                       Buffer);
```

```
    /*
    Set VGA color register 65 to red, 66 to green and 67 to
    blue so the histogram can be visually separated from
    the continuous tone image.
    */


    SetAColorReg(65,63,0,0);
    SetAColorReg(66,0,63,0);
    SetAColorReg(67,0,0,63);

    /* Calculate the histogram for the image */
    GenHistogram(ImageData, Col, Row, Width, Height);

    MaxRepeat = 0;

    /*
    Find the pixel value repeated the most. It will be used for
    scaling.
    */
    for (Index=0; Index < MAXQUANTLEVELS; Index++)
      MaxRepeat = (Histogram[Index] > MaxRepeat) ?
                Histogram[Index]:MaxRepeat;


    /* Fill background area of histogram graph */
    ClearImageArea(ImageData,HISTOCOL,HISTOROW,HISTOWIDTH,HISTOHEIGHT,67);

    /* Draw the bounding box for the histogram */
    DrawVLine(ImageData,HISTOCOL,HISTOROW,HISTOHEIGHT-1,BLACK);
    DrawVLine(ImageData,HISTOCOL+HISTOWIDTH-1,HISTOROW,HISTOHEIGHT-1,BLACK);
    DrawHLine(ImageData,HISTOCOL,HISTOROW+HISTOHEIGHT-1,HISTOWIDTH-1,BLACK);
    DrawHLine(ImageData,HISTOCOL,HISTOROW,HISTOWIDTH-1,BLACK);

    /* Data base line */
    DrawHLine(ImageData,AXISCOL,AXISROW,AXISLENGTH,WHITE);
    DrawHLine(ImageData,AXISCOL,AXISROW+1,AXISLENGTH,WHITE);
    /*
    Now do the actual histogram rendering into the
    image buffer.
    */
    for (Index=0; Index < MAXQUANTLEVELS; Index++)
    {
      LineLength = (unsigned)(((long) Histogram[Index] * MAXDEFLECTION) /
                              (long) MaxRepeat);
      XPos = DATACOL + Index*2;
      YPos = DATAROW - LineLength;
      DrawVLine(ImageData,XPos,YPos,LineLength,66);
    }

    /*
    Display the image overlayed with the histogram
    */
    DisplayImageInBuf(ImageData,NOVGAINIT,WAITFORKEY);

    /* After display, restore image data under histogram */
    WriteImageAreaFromBuf(Buffer,HISTOWIDTH,HISTOHEIGHT,ImageData,
                          HISTOCOL,HISTOROW);
    farfree((BYTE far *)Buffer);
}
```

```
/* Various Point Transformation Functions */


void AdjImageBrightness(BYTE huge *ImageData, short BrightnessFactor,
                        unsigned Col, unsigned Row,
                        unsigned Width, unsigned Height)
{
  register unsigned Index;
  register short NewLevel;
  BYTE   LookUpTable[MAXQUANTLEVELS];

  for (Index = MINSAMPLEVAL; Index < MAXQUANTLEVELS; Index++)
  {
    NewLevel = Index + BrightnessFactor;
    NewLevel = (NewLevel < MINSAMPLEVAL) ? MINSAMPLEVAL:NewLevel;
    NewLevel = (NewLevel > MAXSAMPLEVAL) ? MAXSAMPLEVAL:NewLevel;
    LookUpTable[Index] = NewLevel;
  }
  PtTransform(ImageData,Col,Row,Width,Height,LookUpTable);
}



/*
This function will negate an image pixel by pixel. Threshold is
the value of image data where the negatation begins. If
threshold is 0, all pixel values are negated. That is, pixel value 0
becomes 63 and pixel value 63 becomes 0. If threshold is greater
than 0, the pixel values in the range 0..Threshold-1 are left
alone while pixel values between Threshold..63 are negated.
*/



void NegateImage(BYTE huge *ImageData, unsigned Threshold,
          unsigned Col, unsigned Row,
                unsigned Width, unsigned Height)
{
  register unsigned Index;
  BYTE   LookUpTable[MAXQUANTLEVELS];

  /* Straight through mapping initially */
  InitializeLUT(LookUpTable);

  /* from Threshold onward, negate entry in LUT */
  for (Index = Threshold; Index < MAXQUANTLEVELS; Index++)
       LookUpTable[Index] = MAXSAMPLEVAL - Index;

  PtTransform(ImageData,Col,Row,Width,Height,LookUpTable);
}



/*
This function converts a gray scale image to a binary image with each
pixel either on (WHITE) or off (BLACK). The pixel level at
which the cut off is made is controlled by Threshold. Pixels
in the range 0..Threshold-1 become black while pixel values
between Threshold..63 become white.
*/
```

```
void ThresholdImage(BYTE huge *ImageData, unsigned Threshold,
                    unsigned Col, unsigned Row,
                        unsigned Width, unsigned Height)
{
  register unsigned Index;
  BYTE    LookUpTable[MAXQUANTLEVELS];

  for (Index = MINSAMPLEVAL; Index < Threshold; Index++)
        LookUpTable[Index] = BLACK;

  for (Index = Threshold; Index < MAXQUANTLEVELS; Index++)
        LookUpTable[Index] = WHITE;

  PtTransform(ImageData,Col,Row,Width,Height,LookUpTable);
}


void StretchImageContrast(BYTE huge *ImageData, unsigned *HistoData,
                          unsigned Threshold,
                    unsigned Col, unsigned Row,
                          unsigned Width, unsigned Height)
{
  register unsigned Index, NewMin, NewMax;
  double  StepSiz, StepVal;
  BYTE    LookUpTable[MAXQUANTLEVELS];

  /*
  Search from the low bin towards the high bin for the first one that
  exceeds the threshold
  */

  for (Index=0; Index < MAXQUANTLEVELS; Index++)
    if (HistoData[Index] > Threshold)
      break;

  NewMin = Index;

  /*
  Search from the high bin towards the low bin for the first one that
  exceeds the threshold
  */

  for (Index=MAXSAMPLEVAL; Index > NewMin; Index--)
    if (HistoData[Index] > Threshold)
      break;

  NewMax = Index;

  StepSiz = (double)MAXQUANTLEVELS/(double)(NewMax-NewMin+1);
  StepVal = 0.0;

  /* values below new minimum are assigned zero in the LUT */
  for (Index=0; Index < NewMin; Index++)
    LookUpTable[Index] = MINSAMPLEVAL;

  /* values above new maximum are assigned the max sample value */
  for (Index=NewMax+1; Index < MAXQUANTLEVELS; Index++)
    LookUpTable[Index] = MAXSAMPLEVAL;

  /* values between the new minimum and new maximum are stretched */
  for (Index=NewMin; Index <= NewMax; Index++)
  {
```

```
    LookUpTable[Index] = StepVal;
    StepVal += StepSiz;
  }
  /*
  Look Up Table is now prepared to point transform the image data.
  */
  PtTransform(ImageData,Col,Row,Width,Height,LookUpTable);
}
```

P A T E N T O V É N Á R O K Y

1. Způsob vytváření trojrozměrného obrazu ze zobrazení dvojrozměrného obrazu, vytvořeného diskrétními obrazovými prvky, obsahující zajištění soustavy optických prvků, příslušně scentrovaných před obrazovými prvky, a měnění efektivní ohniskové vzdálenosti každého optického prvku pro změnu zdánlivé vizuální vzdálenosti od pozorovatele, nacházejícího se před zobrazovačem, na němž je znázorněn každý jednotlivý obrazový bod, v y z n a č u j í c í s e t í m , že každý optický prvek má ohniskovou délku, která se mění progresivně podél ploch orientovaných obecně rovnoběžně s obrazem, přičemž měnění efektivní ohniskové délky každého optického prvku obsahuje kroky posuvu místa, na němž je světlo emitováno z dvourozměrného obrazu bezprostředně uvnitř každého obrazového prvku, a převádění emitovaného světla k optickým prvkům, kde místo, na nějž emitované světlo na optických prvcích dopadá, určuje zdánlivou hloubku obrazového bodu.

2. Způsob vytváření trojrozměrného obrazu podle nároku 1, v y z n a č u j í c í s e t í m , že optické prvky jsou lomové prvky a světlo vstupuje do lomové plochy přidruženého lomového prvku.

3. Způsob vytváření trojrozměrného obrazu podle nároku 1, v y z n a č u j í c í s e t í m , že optické prvky jsou

zrcadla a světlo dopadá na odrazovou plochu přidruženého zrcadla.

4. Způsob vytváření trojrozměrného obrazu podle nároku 1, 2 nebo 3, v y z n a č u j í c í s e t í m , že krok posuvu místa, na němž je světlo emitováno z dvourozměrného obrazu, obsahuje lineární posuv bodu, na němž je světlo emitováno z dvourozměrného obrazu.

5. Způsob vytváření trojrozměrného obrazu podle nároku 1, 2 nebo 3, v y z n a č u j í c í s e t í m , že krok zobrazení místa, na němž je světlo emitováno z dvourozměrného obrazu, obsahuje radiální posuv místa, na němž je světlo emitováno z dvourozměěného obrazu.

6. Zobrazovací zařízení pro provádění způsobu podle kteréhokoliv z nároků 1 až 5, na němž je znázorněn obraz vytvořený diskrétními obrazovými body, kde toto zařízení je opatřeno soustavou optických prvků, příslušně zcentrovaných před obrazovými body, a prostředky pro individuální měnění efektivní ohniskové délky každého optického prvku pro změnu zdánlivé vizuální vzdálenosti od pozorovatele, nacházejícího se před zobrazovacím zařízením, na němž se zobrazuje každý jednotlivý obrazový bod, čímž se vytváří trojrozměrný obraz, v y z n a č u j í c í s e t í m , že je opatřeno prostředky (18, 65) pro malý posuv místa (5b, 6b, 7b), v němž je světlo v obrazovém bodě emitováno, a to podle požadované hloubky tak, že je zde odpovídající posuv místa

(5, 6, 7) vstupu světla podél vstupní plochy optického prvku, čímž se účinná ohniskové délka dynamicky mění a zdánlivá vizuální vzdálenost (5a, 6a, 7a) od pozorovatele se mění podle posuvu místa vstupu světla.

7. Zobrazovací zařízení podle nároku 6, v y z n a č u j í c í s e t í m , že optické prvky (2) jsou lomové prvky a vstupní plocha je lomová plocha.

8. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že lomové plochy jsou tvarovány pro zajištění proměnné ohniskové délky.

9. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že optické lomové prvky (2) jsou vyrobeny z optického materiálu s proměnným indexem lomu, v němž se index lomu mění progresivně podél lomového prvku pro vytvoření proměnné ohniskové délky.

10. Zobrazovací zařízení podle nároku 7, 8 nebo 9, v y z n a č u j í c í s e t í m , že vztah mezi posuvem a ohniskovou délkou je lineární.

11. Zobrazovací zařízení podle nároku 7, 8 nebo 9, v y z n a č u j í c í s e t í m , že vztah mezi posuvem a ohniskovou délkou je nelineární.

12. Zobrazovací zařízení podle kteréhokoliv z nároků 7 až 11, v y z n a č u j í c í s e t í m , že každý optický lomový prvek (39) má ohniskovou délku měnící se radiálně vůči optické ose optického lomového prvku a posouvací prostředek posouvá radiálně v obrazovém bodu místo (40a, 41a, 42a), na němž je světlo emitováno.

13. Zobrazovací zařízení podle kteréhokoliv z nároků 7 až 11, v y z n a č u j í c í s e t í m , že každý optický lomový prvek (2) je podlouhlý a má ohniskovou vzdálenost měnící se podél jeho délky od jednoho konce, a zobrazovací prostředek posouvá lineárně v obrazovém bodě bod, v němž je emitováno světlo.

14. Zobrazovací zařízení podle kteréhokoliv z předcházejících nároků, v y z n a č u j í c í s e t í m , že obsahuje jako světelný zdroj jedno zařízení ze skupiny tvořené zobrazovacím zařízením na bázi tekutých krystalů, zobrazovacím zařízením na bázi elektroluminiscence a zobrazovacím zařízením na bázi plasmového zobrazení.

15. Zobrazovací zařízení podle nároku 13, v y z n a č u j í c í s e t í m , že zahrnuje obrazovku (10), opatřenou soustavou podlouhlých fosforových obrazových bodů, a prostředek pro posun místa, na němž je světlo v obrazovém bodě emitováno, obsahuje prostředek (65) pro posuv elektronového svazku podél každého fosforového obrazového bodu.

16. Zobrazovací zařízení podle nároku 15, v y z n a č u j í c í s e t í m , že elektronový svazek (66d) má pravoúhlý průřez.

17. Zobrazovací zařízení podle nároku 15, v y z n a č u j í c í s e t í m , že elektronový svazek (66c) má oválný průřez.

18. Zobrazovací zařízení podle nároku 15, v y z n a č u j í c í s e t í m , že obrazové body jsou uspořádány v řadách, přičemž toto zařízení je tvořeno televizním přijímačem majícím prostředek (58, 59, 61, 62, 63) pro vybrání hloubkové složky pro každý obrazový bod z přijímaného signálu a prostředek (60) pro přidání hloubkového signálu ke konvenčnímu horizontálnímu rozmítanému řádku pro řízení vertikální úrovně horizontálního rozmítaného řádku bod po bodu, čímž se dosáhne stupňovitého rozmítaného řádku (20).

19. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že mezi jednotlivými optickými prvky je vytvořena malá vmezeřená štěrbina.

20. Zobrazovací zařízení podle nároku 19, v y z n a č u j í c í s e t í m , že vmezeřená štěrbina je vyplněna černým neprůhledným materiálem.

21. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že optické prvky jsou vytvořeny ve formě desky vylisované z plastického materiálu.

22. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že optické prvky jsou vytvořeny na desce z plastického materiálu vytvořené injekčním litím.

23. Zobrazovací zařízení podle nároku 7, v y z n a č u j í c í s e t í m , že každý optický prvek je složeným přístrojem obsahujícím alespoň dvě jednotlivé optické složky (Obr. 1(b)).

24. Zobrazovací zařízení podle nároku 23, v y z n a č u j í c í s e t í m , že alespoň dvě individuální optické složky jsou vytvořeny jako alespoň dvě desky vylisované z plastického materiálu, které jsou k sobě přitmeleny.

25. Zobrazovací zařízení podle nároku 23, v y z n a č u j í c í s e t í m , že alespoň dvě jednotlivé optické složky jsou vytvořeny jako alespoň dvě desky vylisované z plastického materiálu, které jsou navzájem spojené u svých okrajů.

26. Zobrazovací zařízení podle nároku 13, v y z n a č u j í c í s e t í m , že je ve formě prohlížečky nebo projektoru pro fotografický film (14) a prostředek pro posuv bodu,

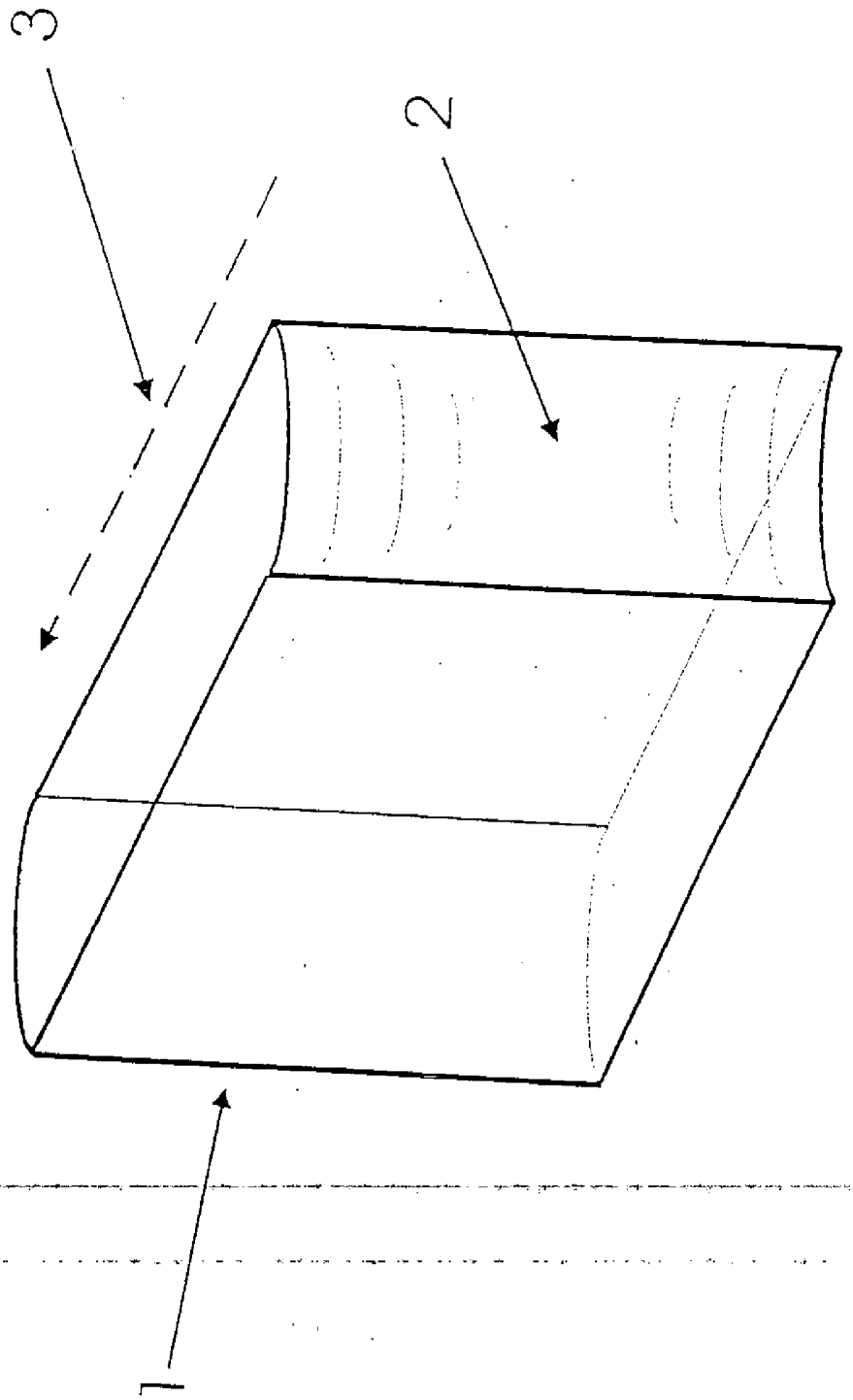
na němž je světlo emitováno, obsahuje masku, přiloženou ke každému obrazovému bodu fotografického filmu tak, že je zajištěn předvolený průsvitný bod (5c).

27. Zobrazovací zařízení podle nároku 6, v y z n a č u j í c í s e t í m , že optické prvky jsou zrcadla (76, 77) a vstupní plocha je odrazová plocha.

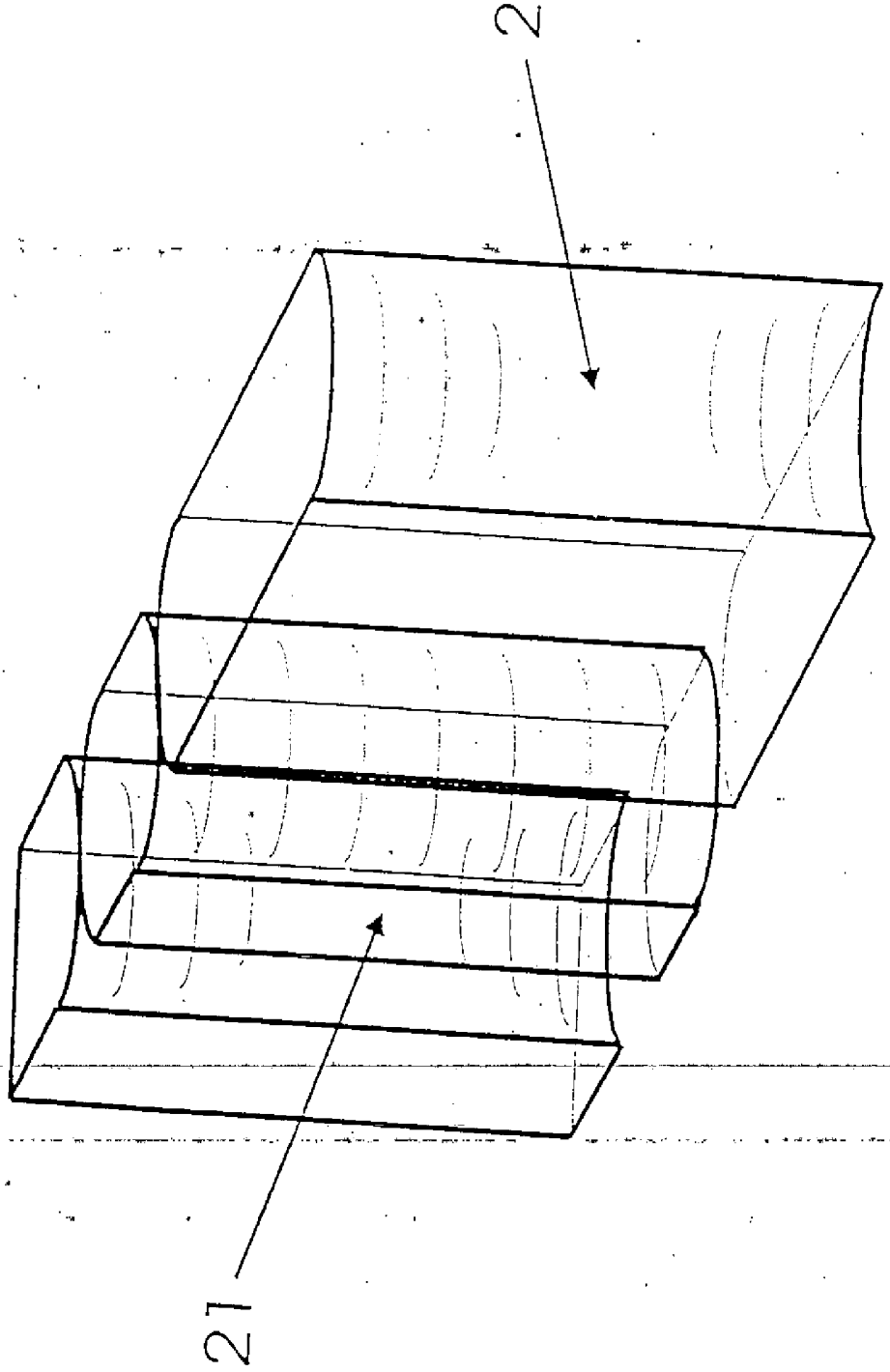
28. Zobrazovací zařízení podle nároku 27 v y z n a č u j í c í s e t í m , že každý optický prvek obsahuje rovinné zrcadlo (76) a konkávní zrcadlo (77).

29. Zobrazovací zařízení podle nároku 28 v y z n a č u j í c í s e t í m , že každé rovinné zrcadlo (76) je vytvořeno jako jedna plocha kombinovaného prvku (78), jejíž další plochu vytváří konkávní zrcadlo (77) sousedního obrazového bodu.

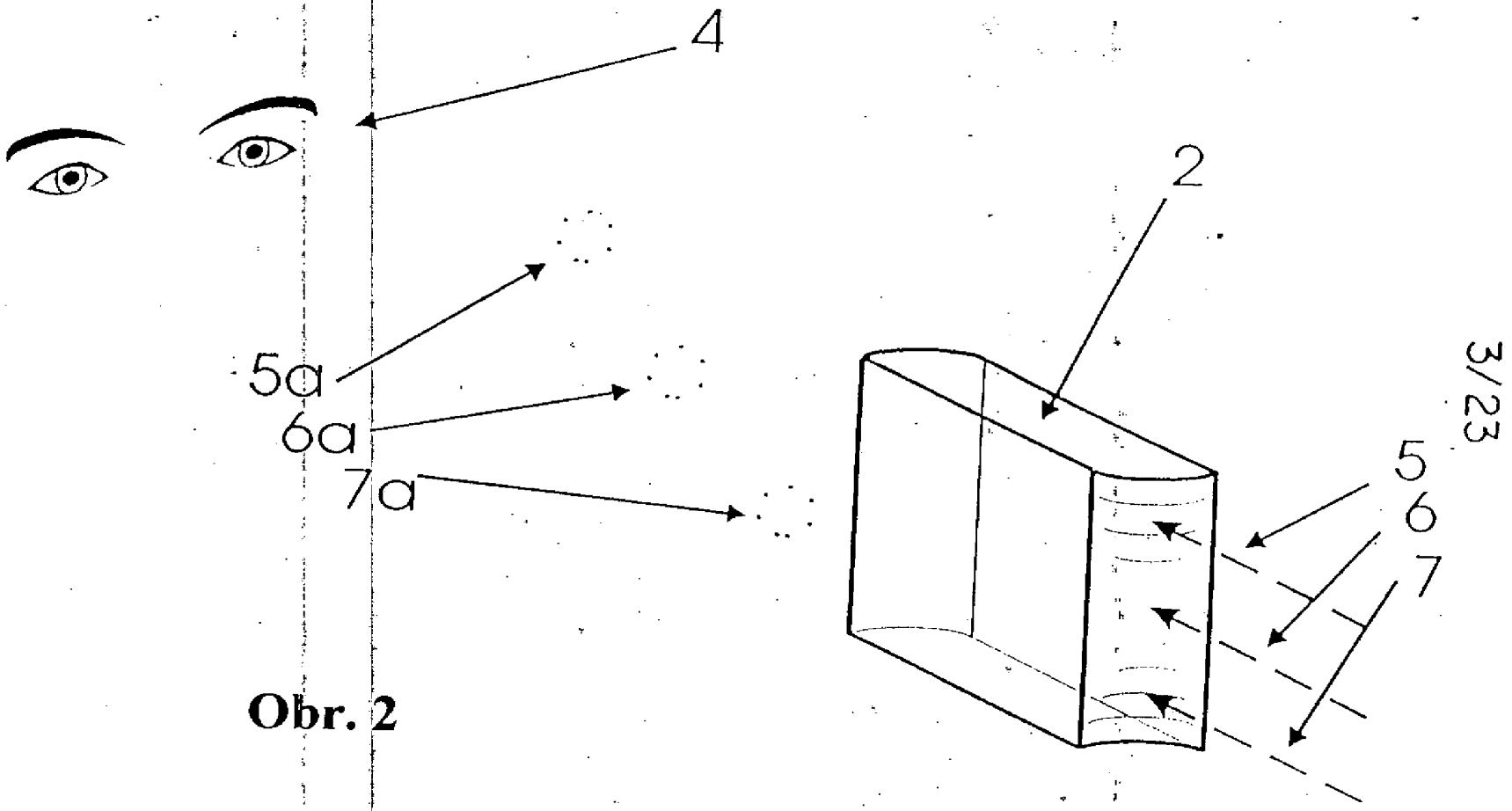
30. Zobrazovací zařízení podle nároku 15, 16 nebo 17, v y z n a č u j í c í s e t í m , že je to počítačový monitor a na počítači založená elektronika obrazového budiče, mající prostředky pro výběr hloubkové složky každého obrazového bodu z dat přijatých z počítače a prostředky (19) pro přidání hloubkové složky k běžnému řádku horizontálního rastru, obrazový bod za obrazovým bodem, čímž je dosaženo stupňového rastru (20).



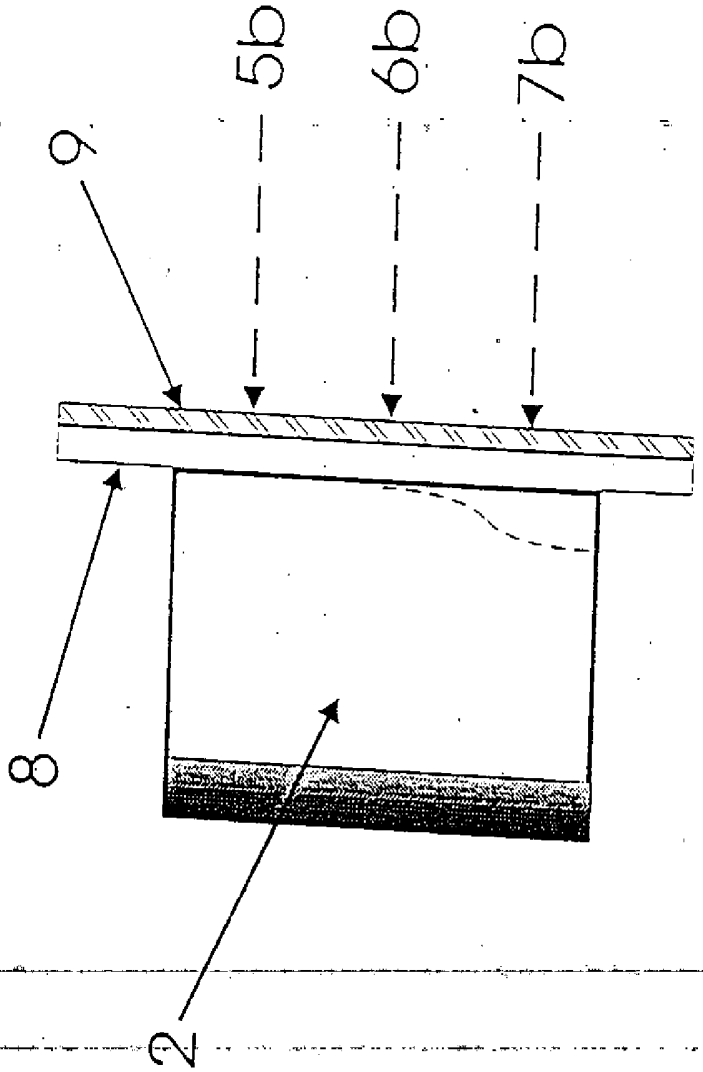
Obr. 1 (a)



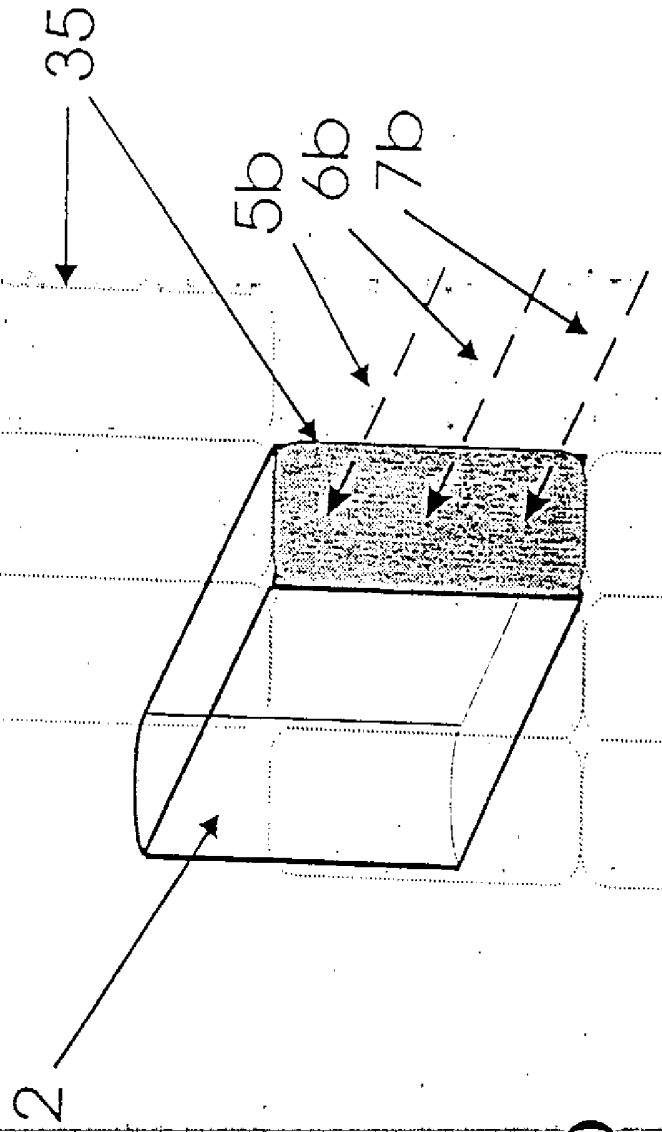
Obr. 1 (b)



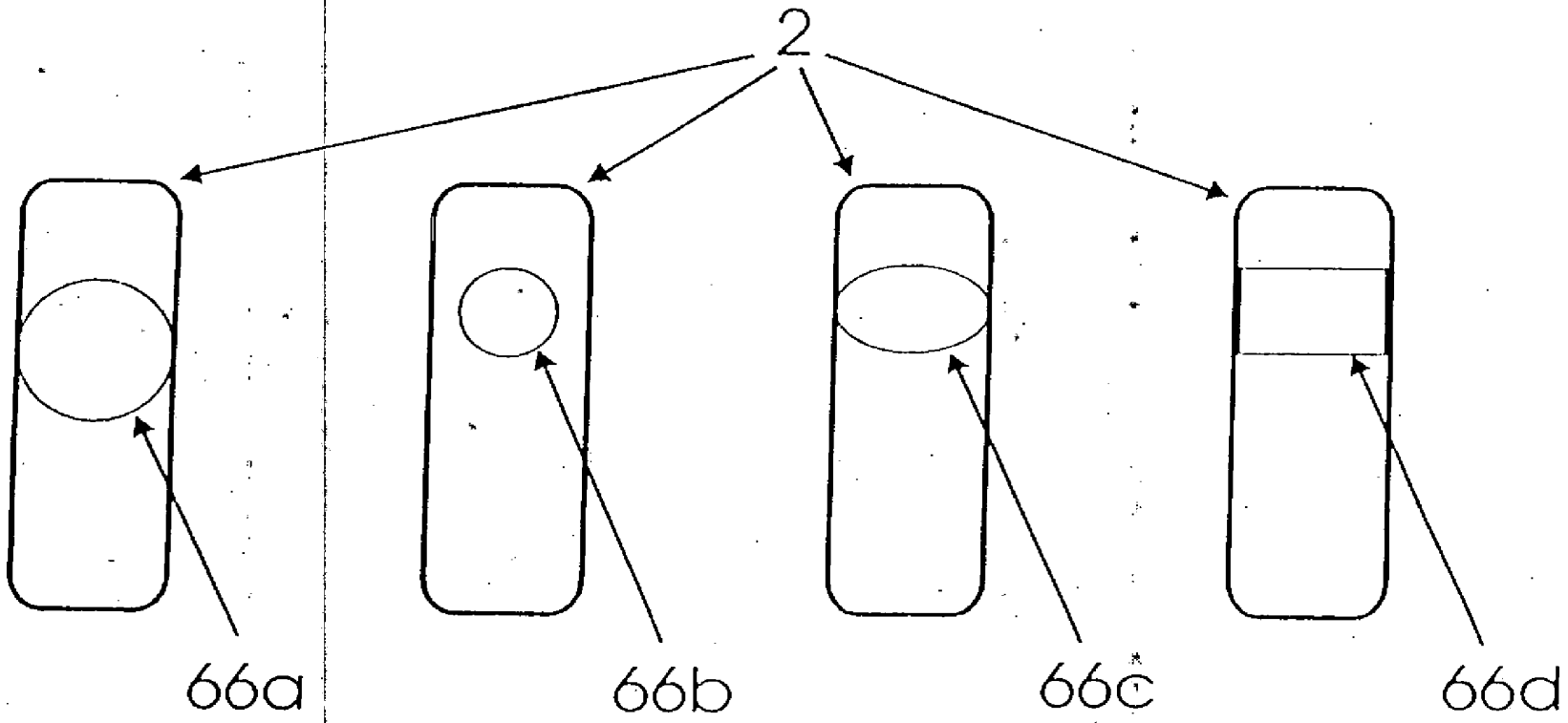
Obr. 2



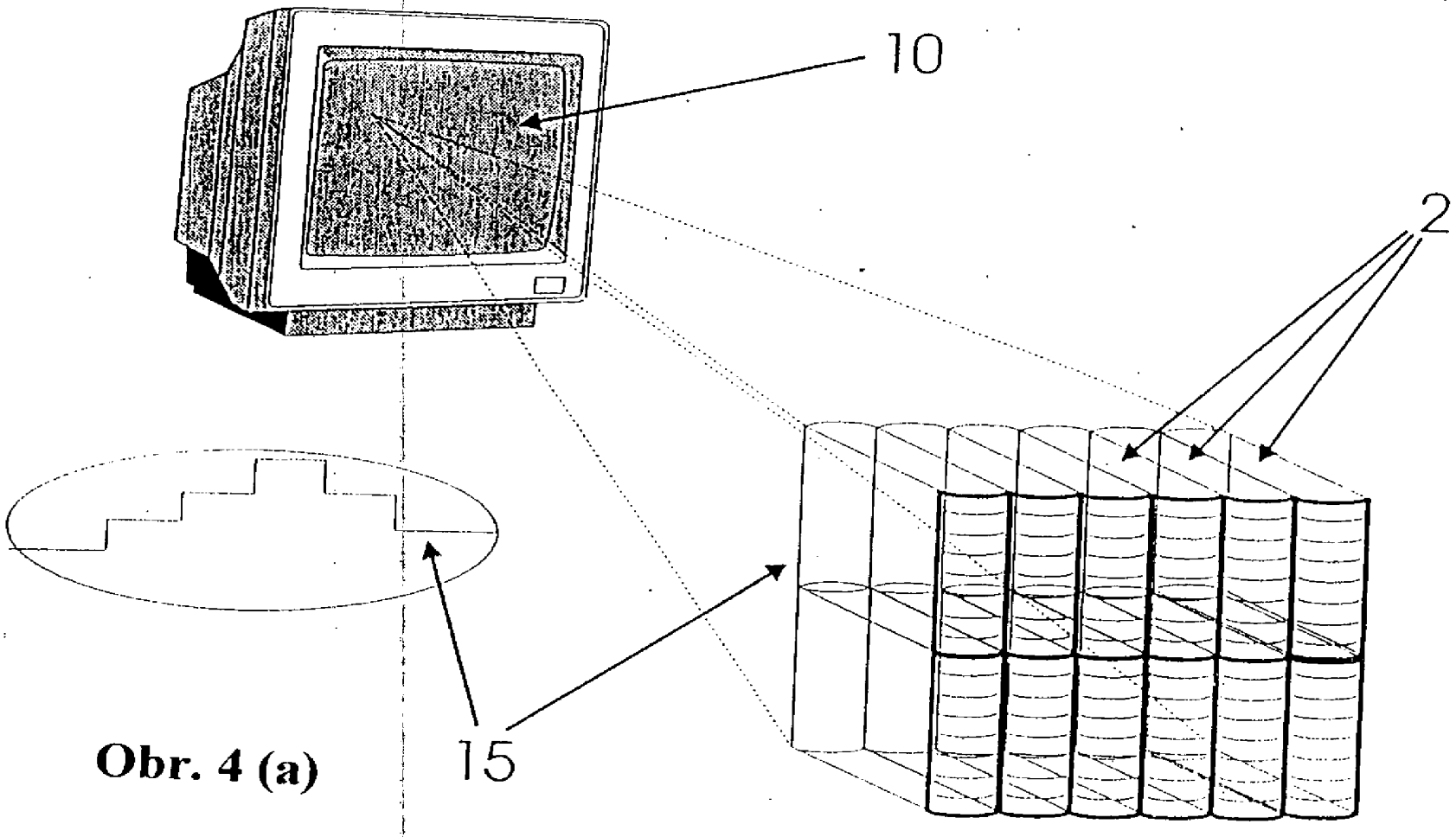
Obr. 3 (a)



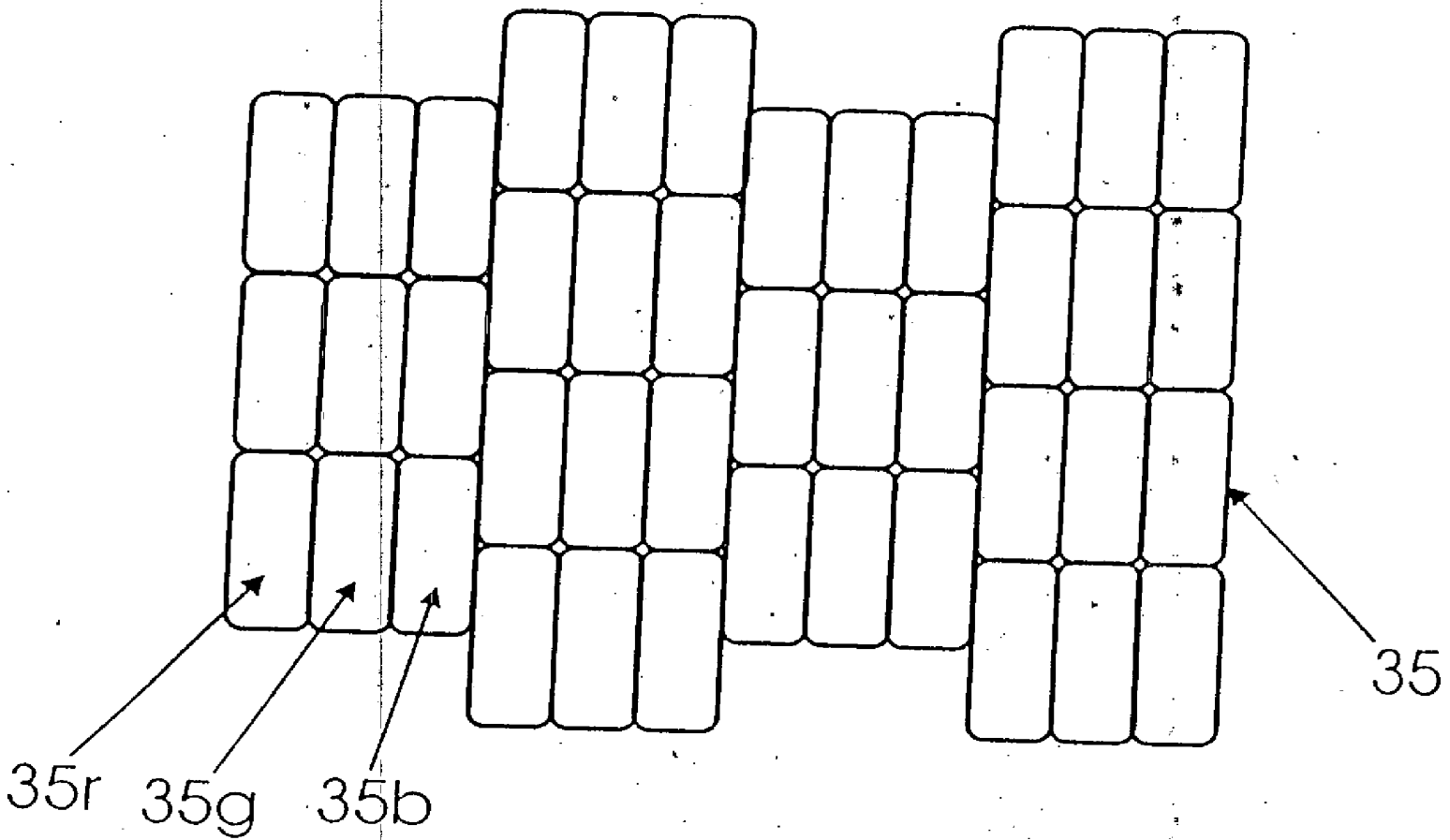
Obr. 3 (b)



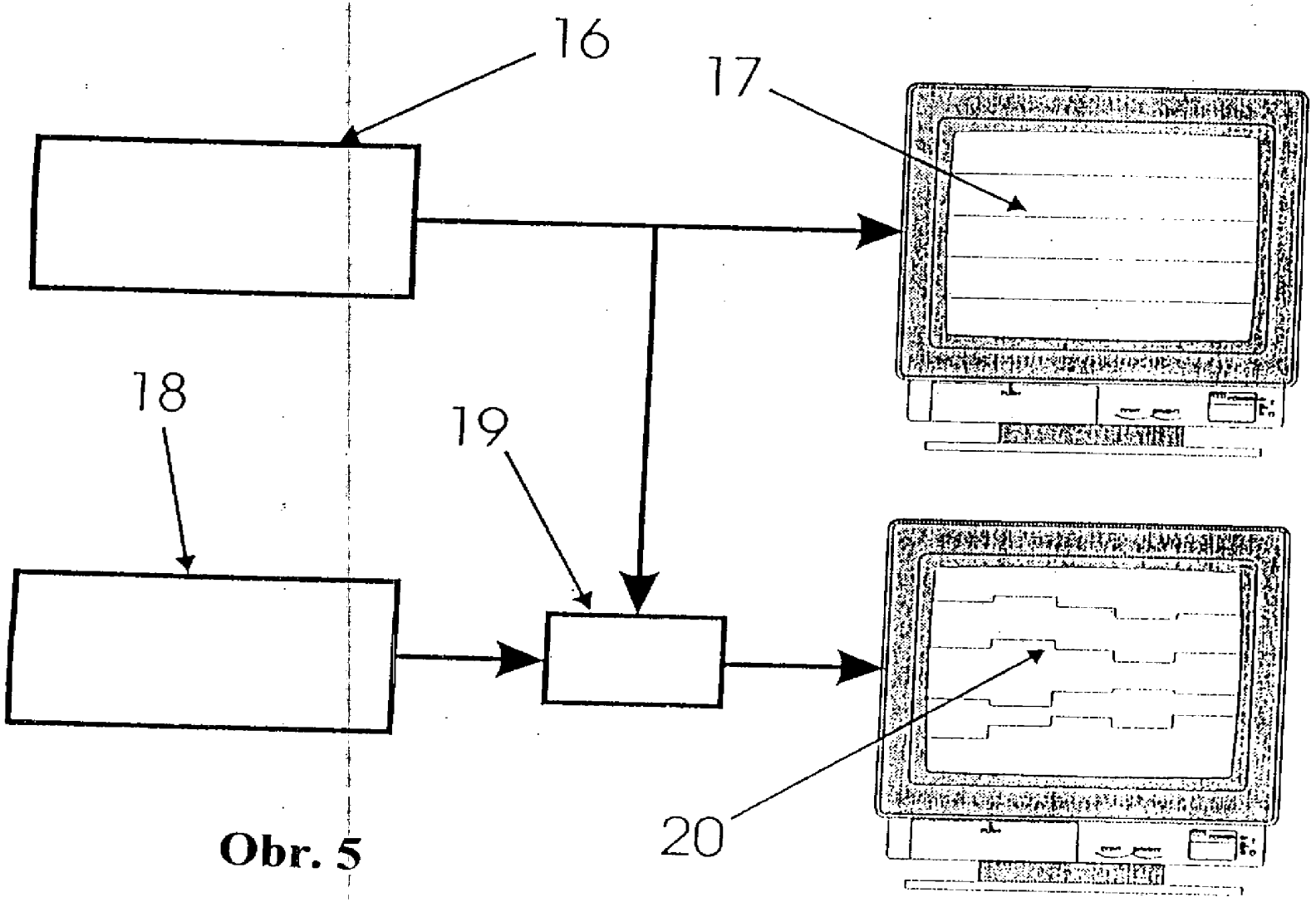
Obr. 3 (c)



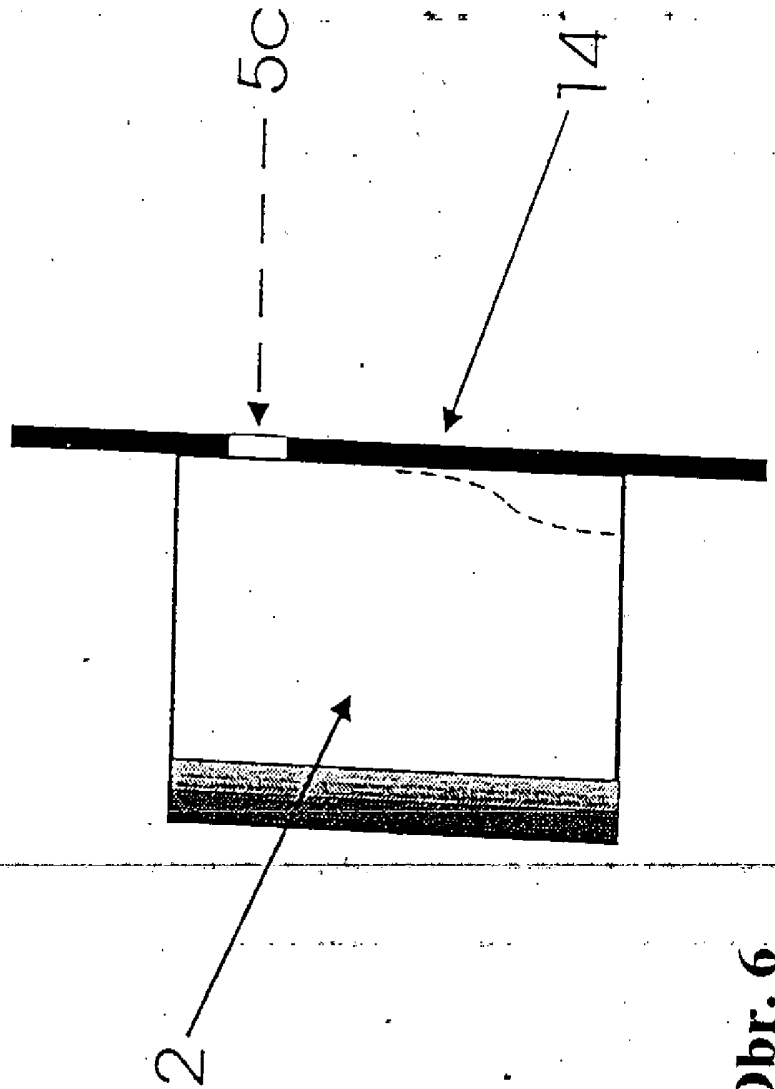
Obr. 4 (a)



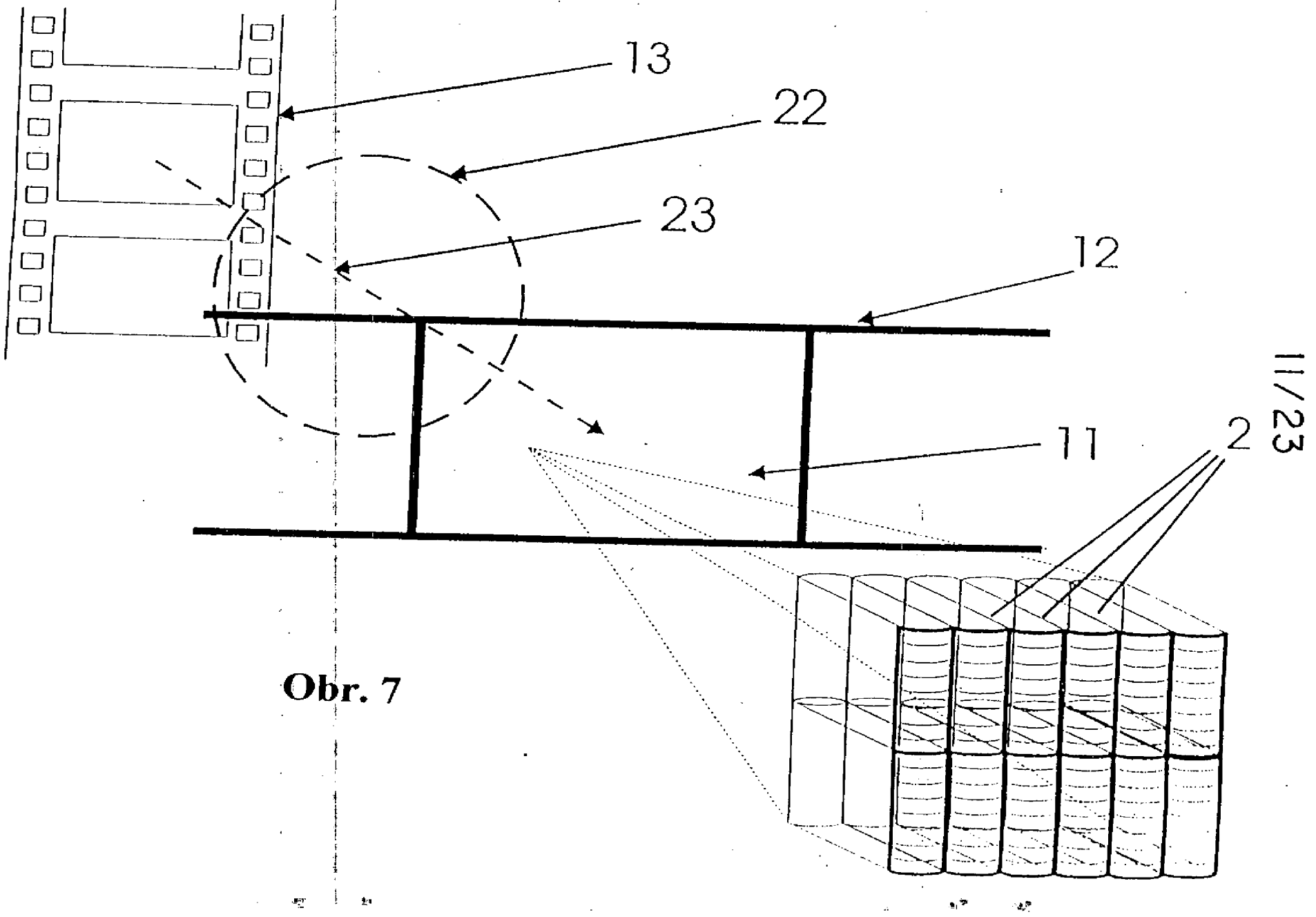
Obr. 4 (b)



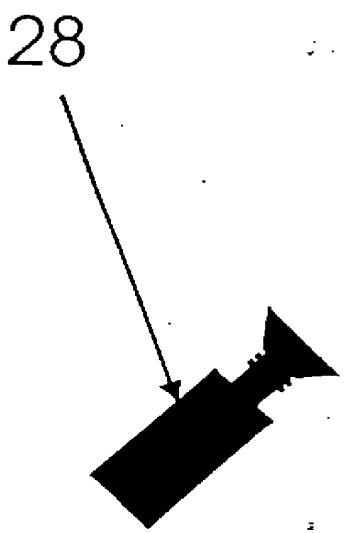
Obr. 5



Obr. 6



Obr. 7

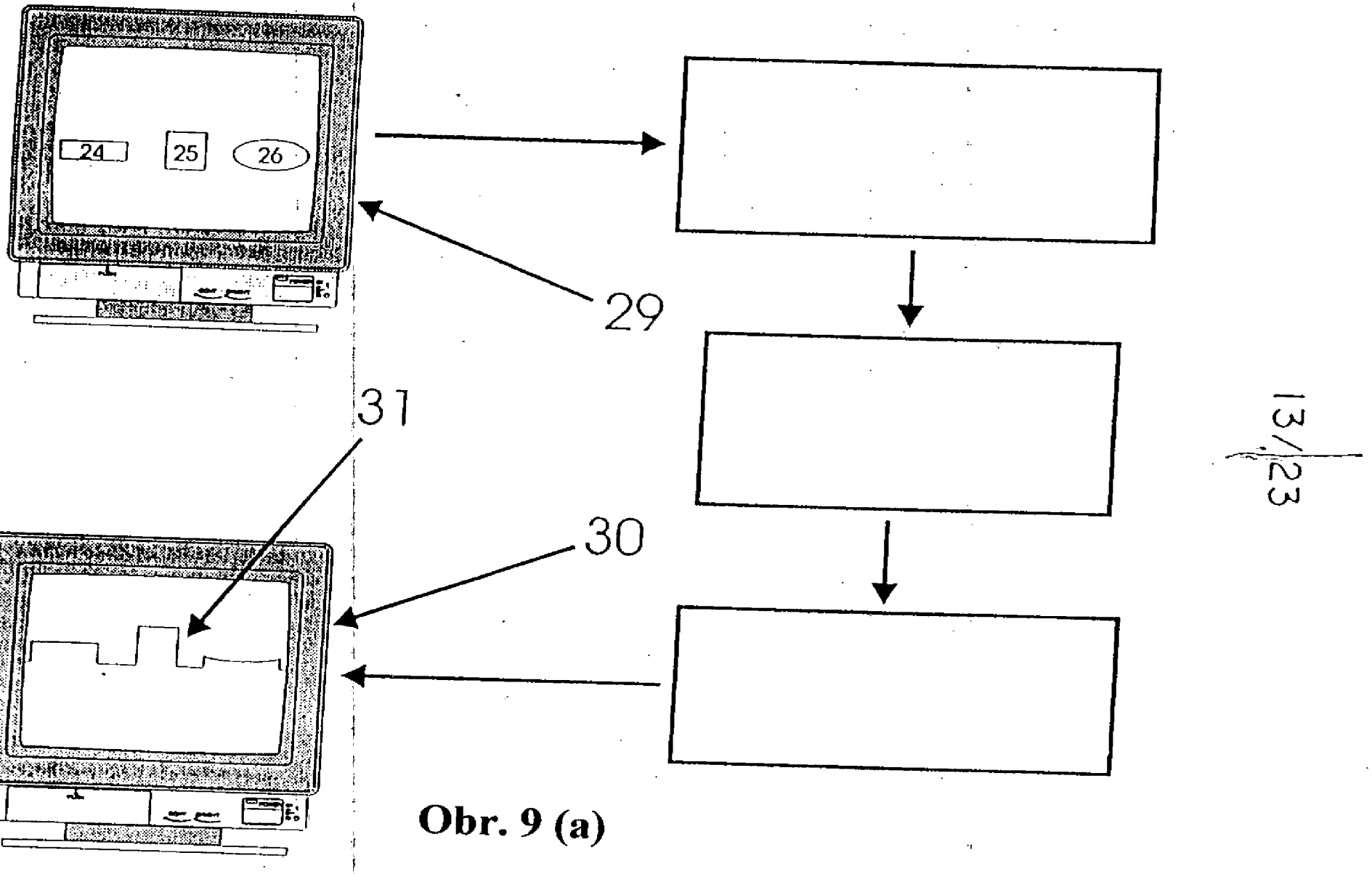


25

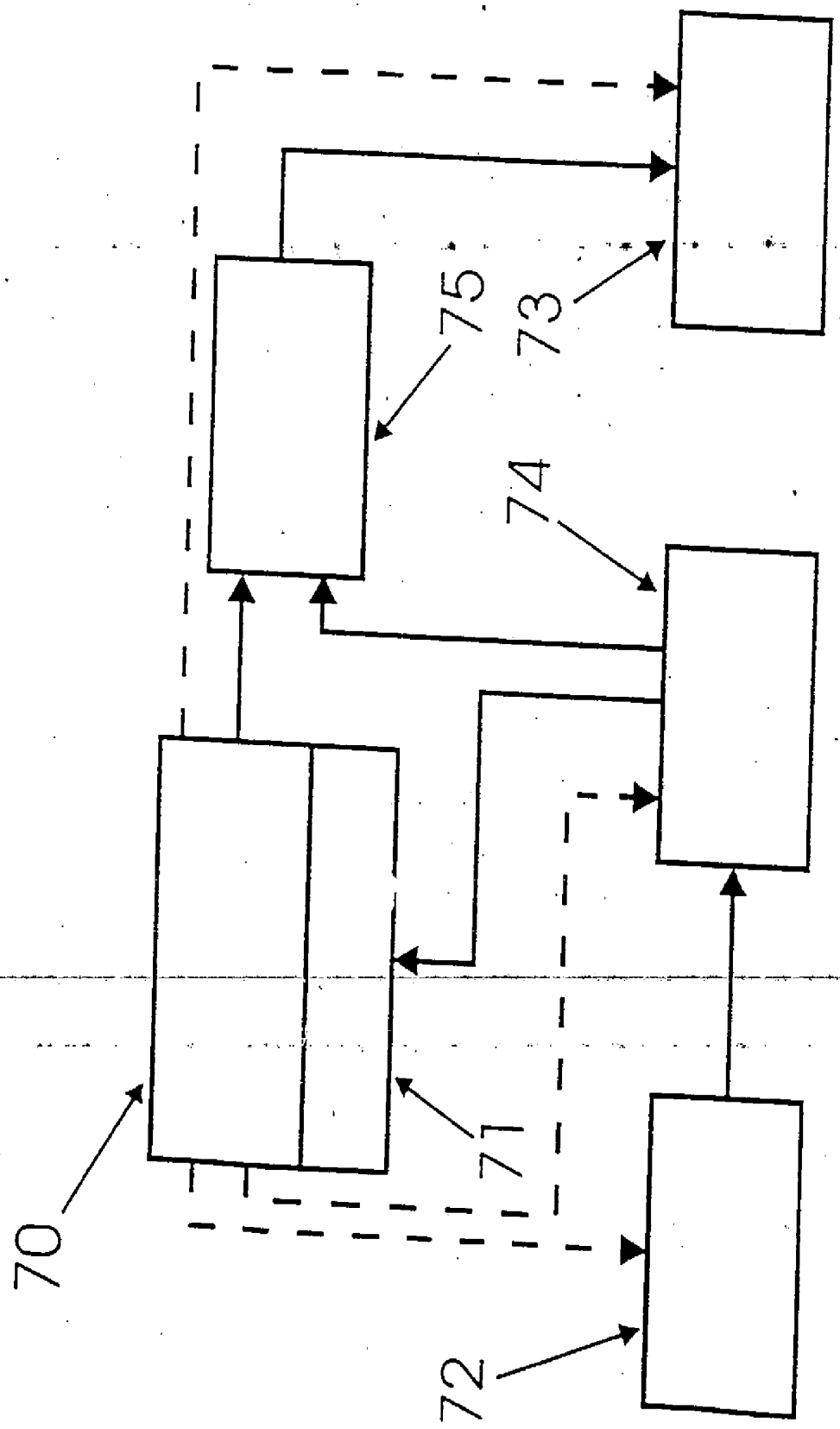
24

26

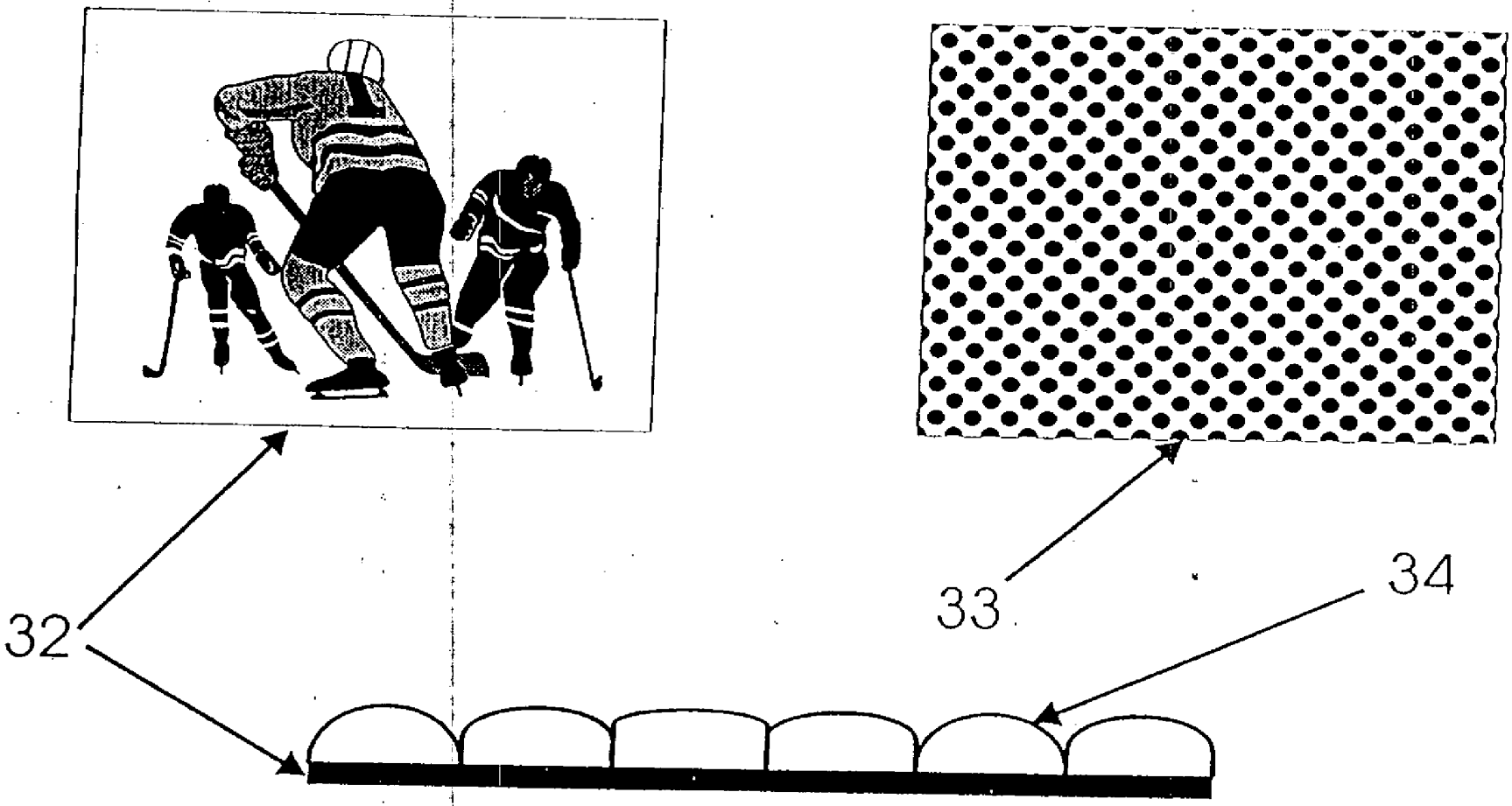
Obr. 8



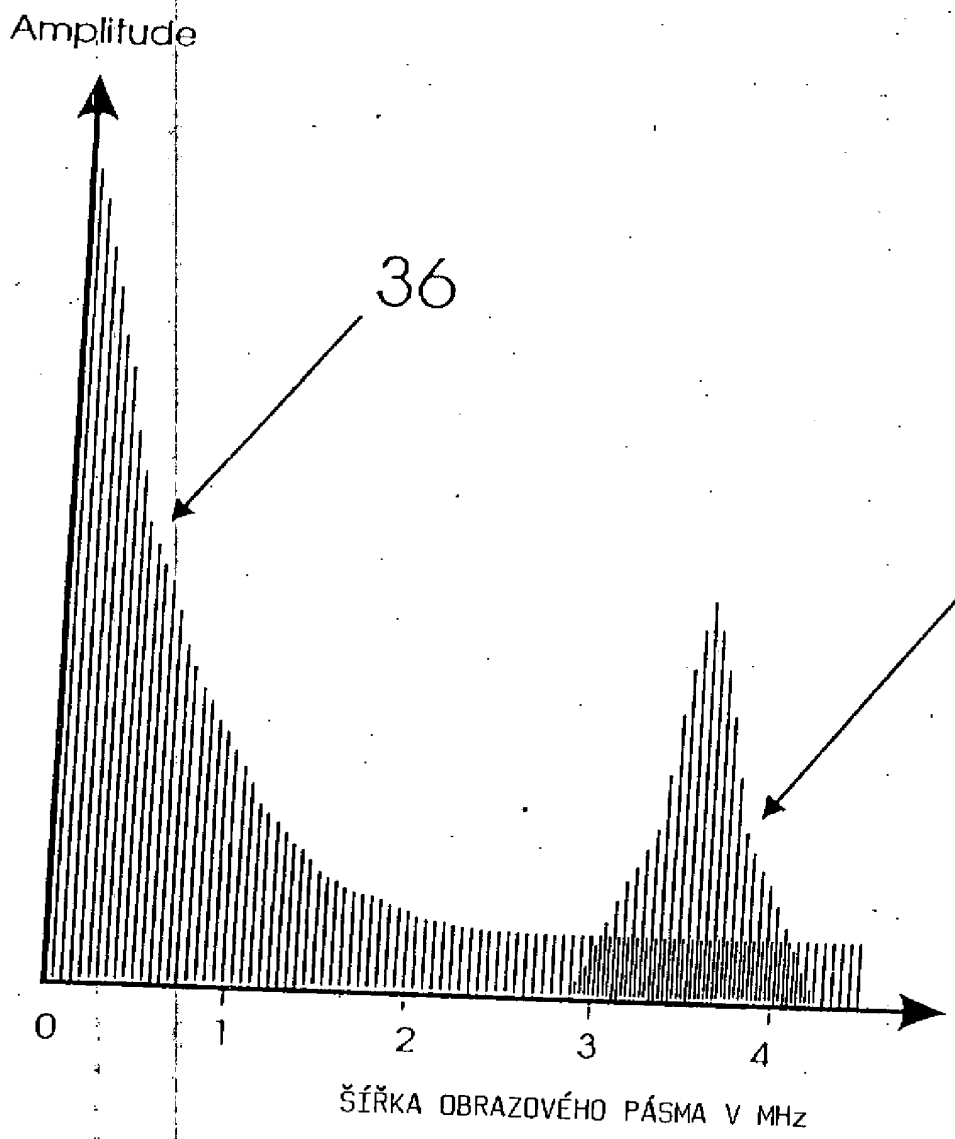
Obr. 9 (a)



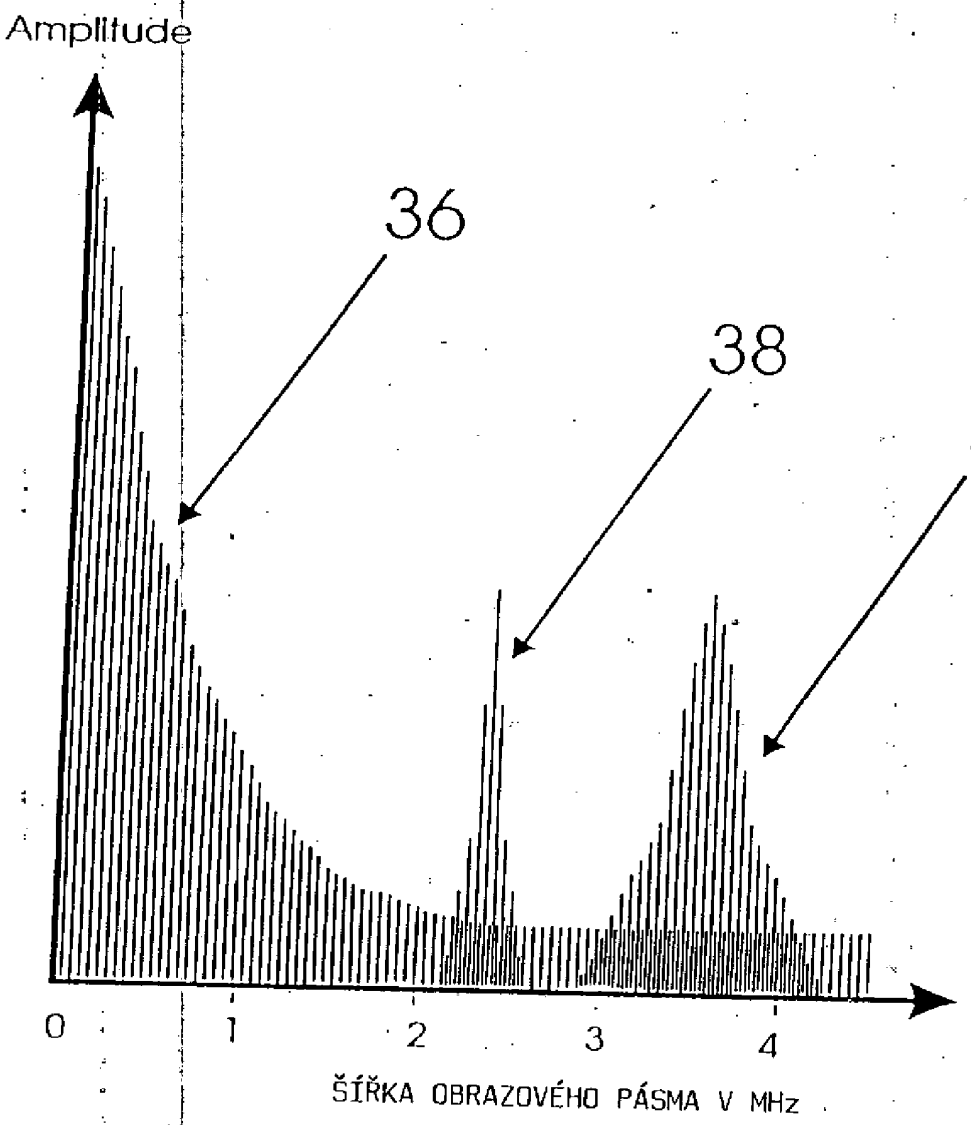
Obr. 9 (b)



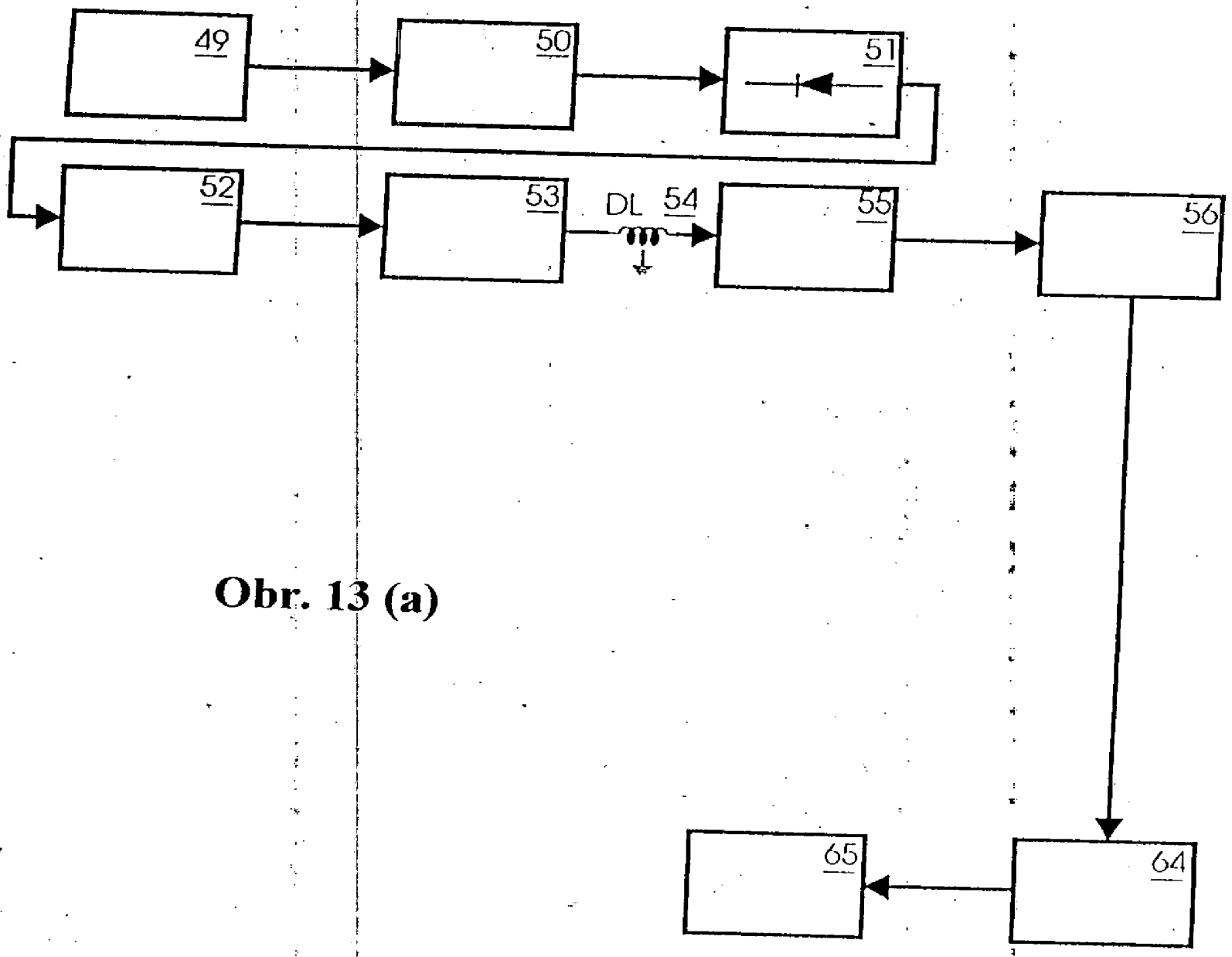
Obr. 10



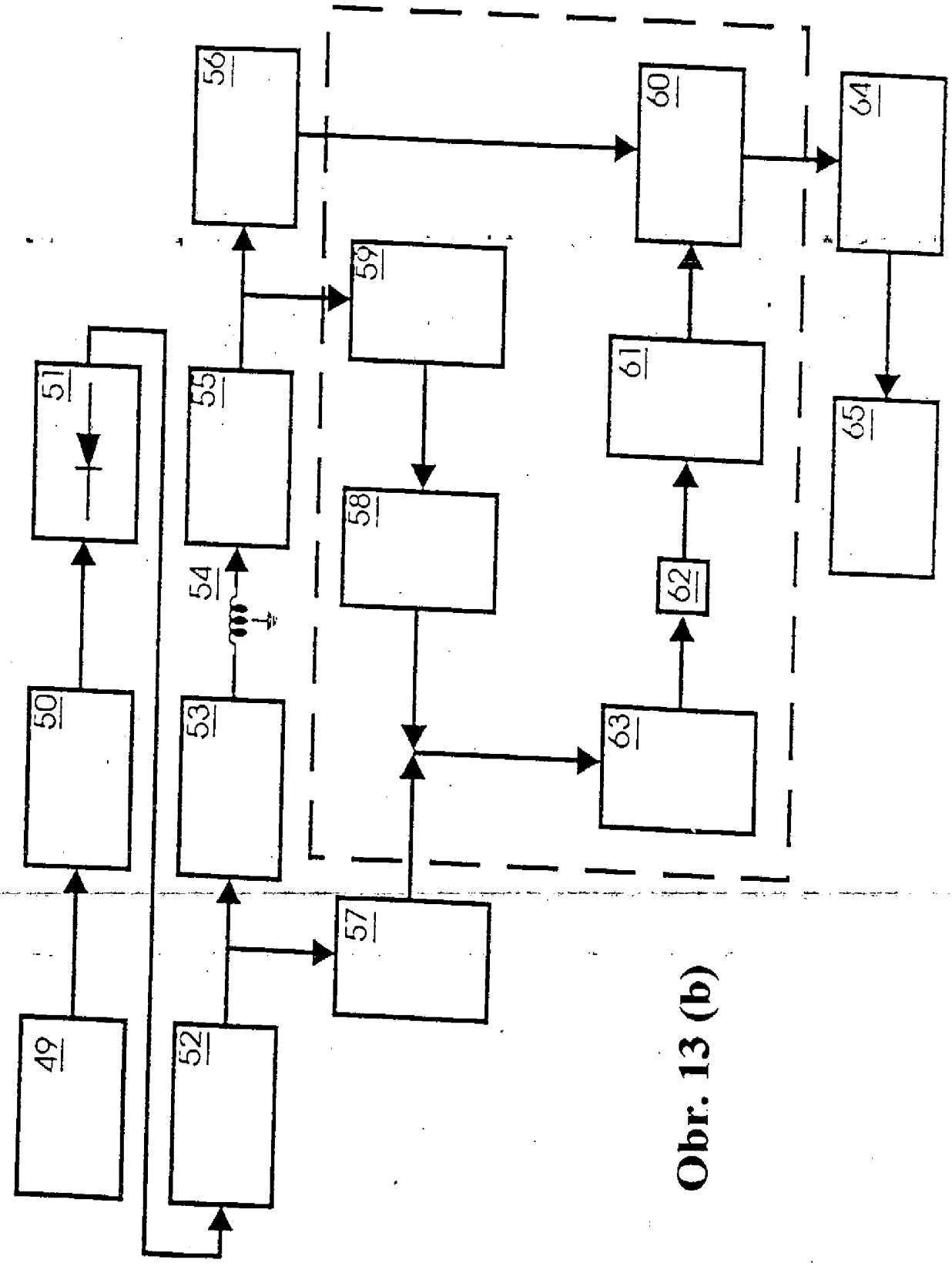
Obr. 11



Obr. 12



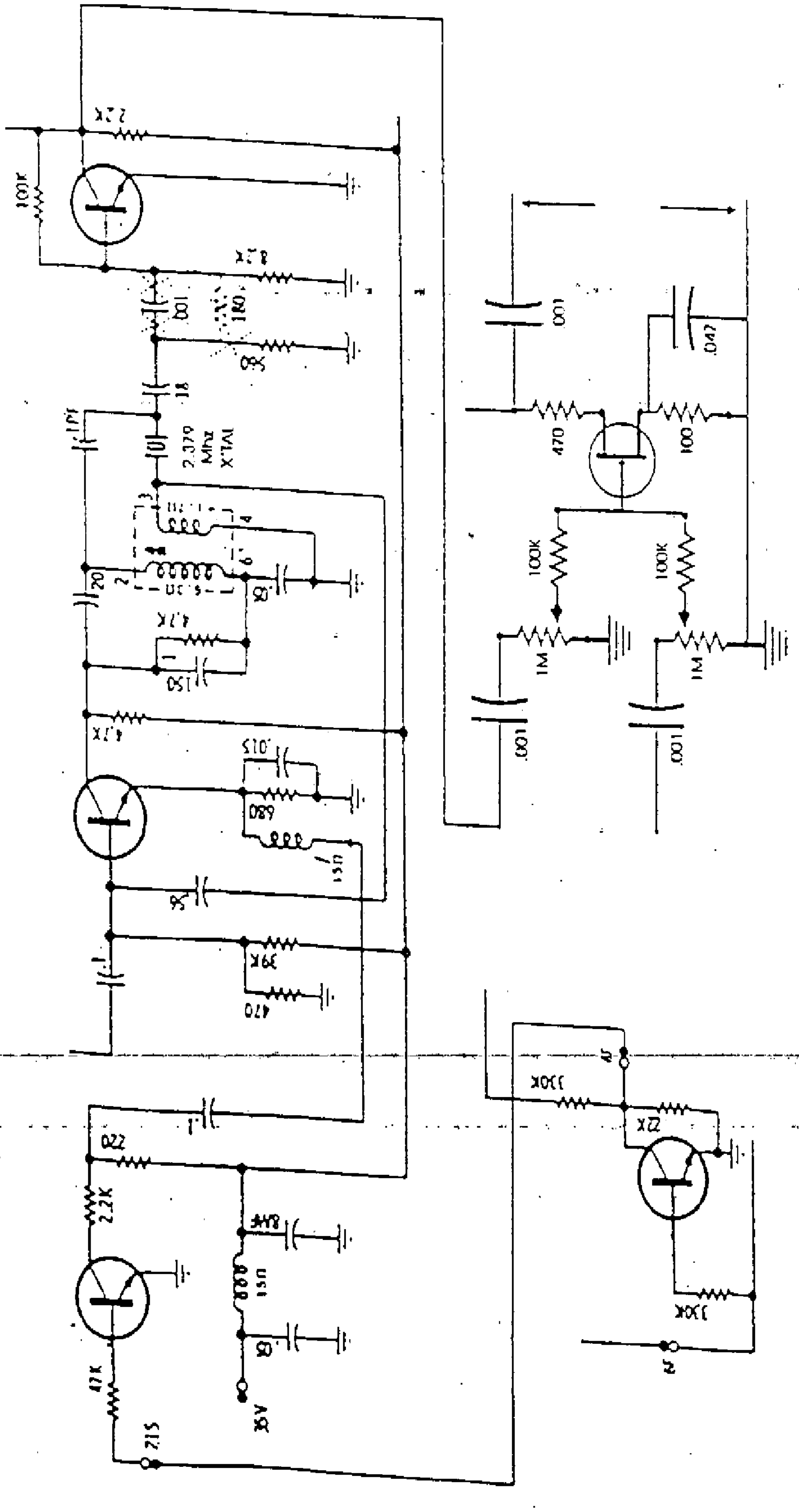
Obr. 13 (a)



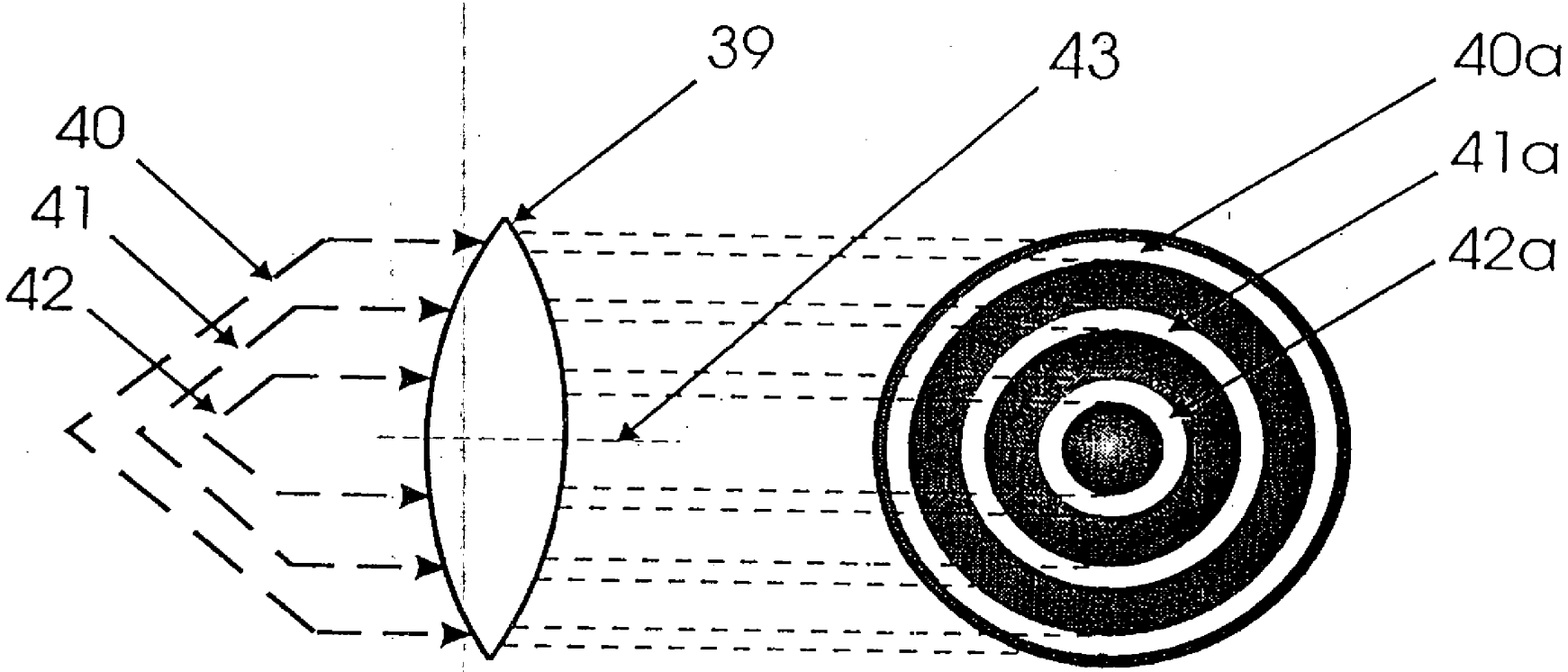
Obr. 13 (b)

16.02.99

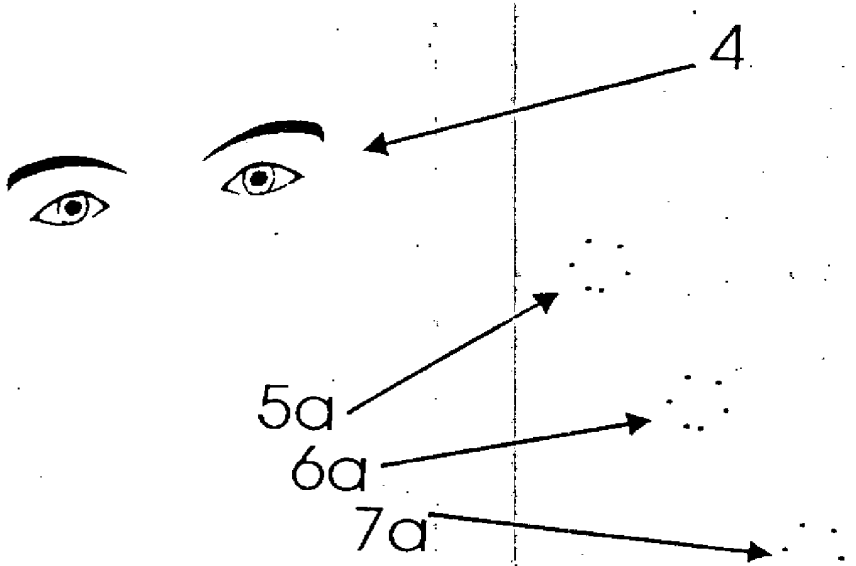
20/23



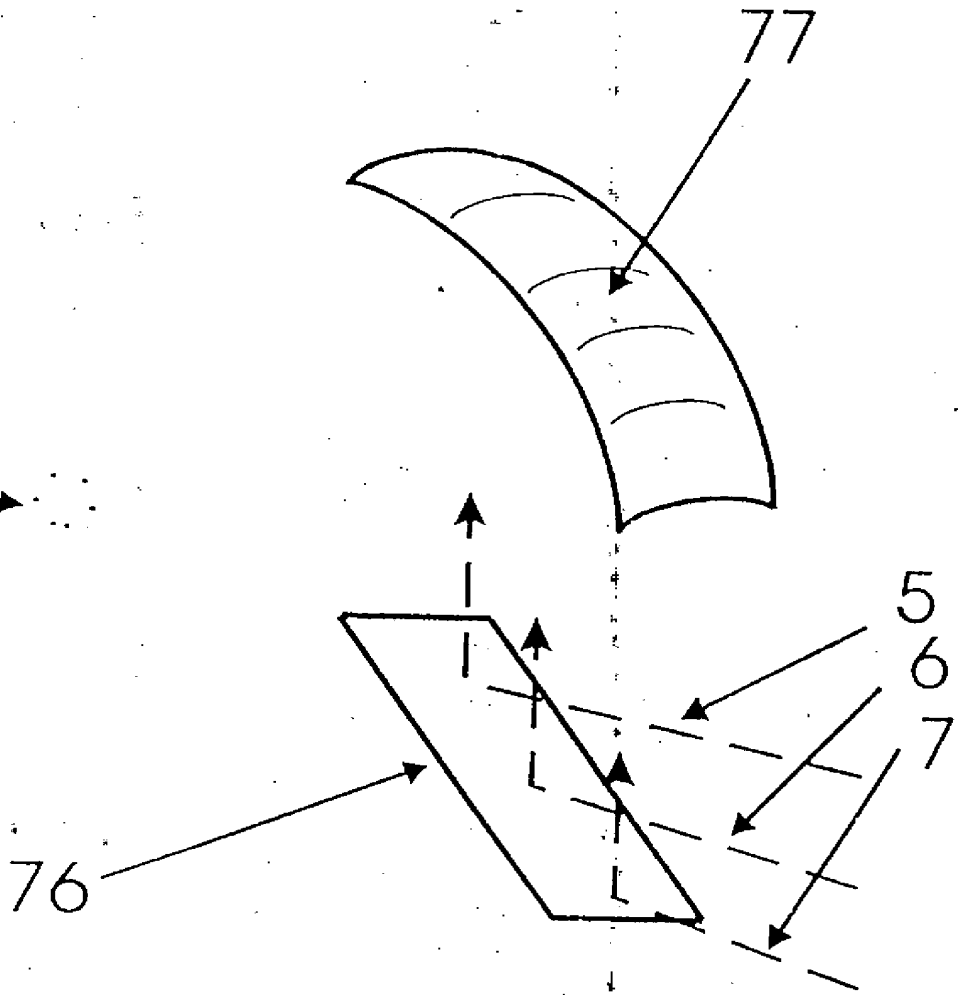
Obr. 14

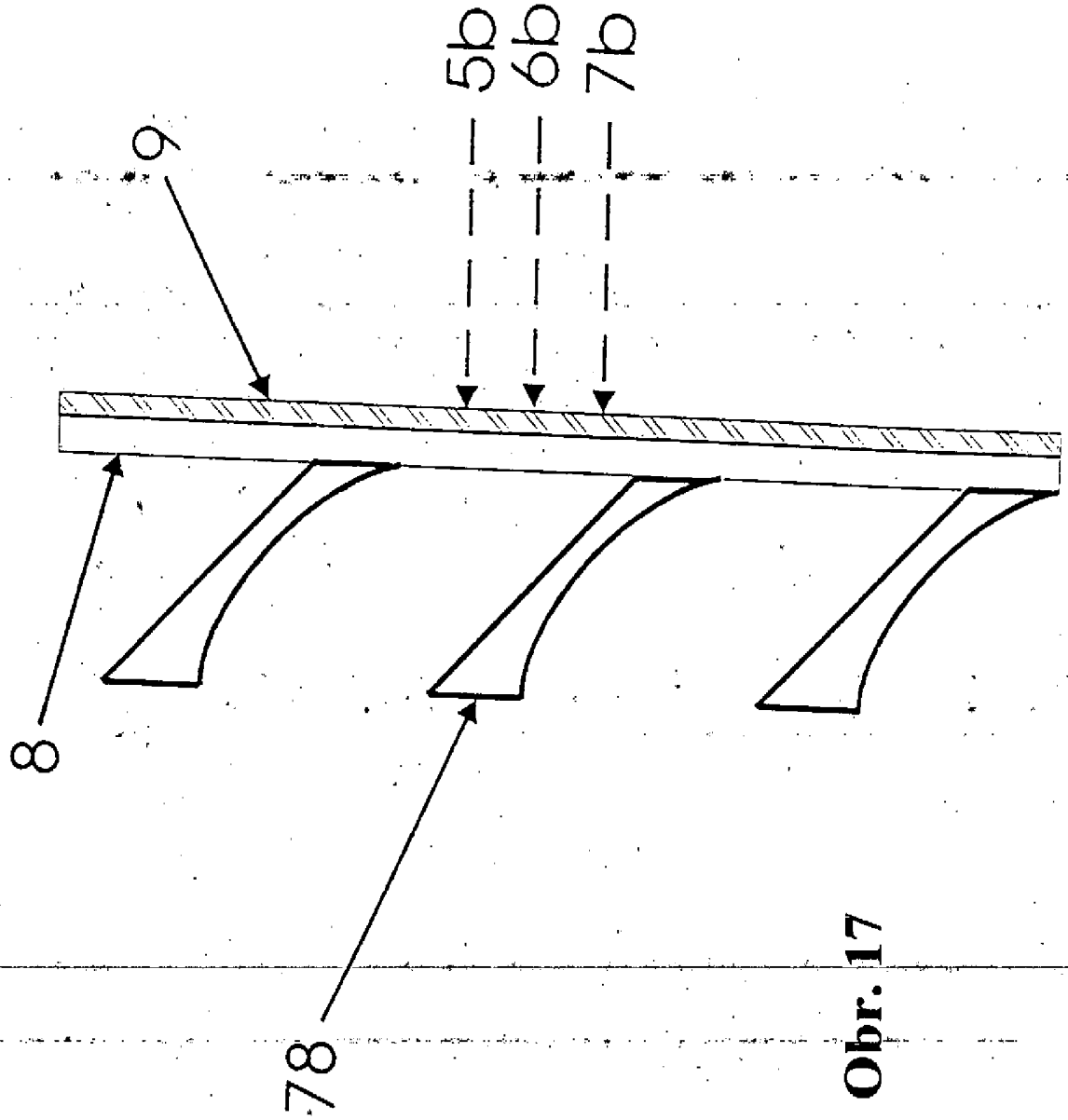


Obr. 15



Obr. 16





Obr. 17